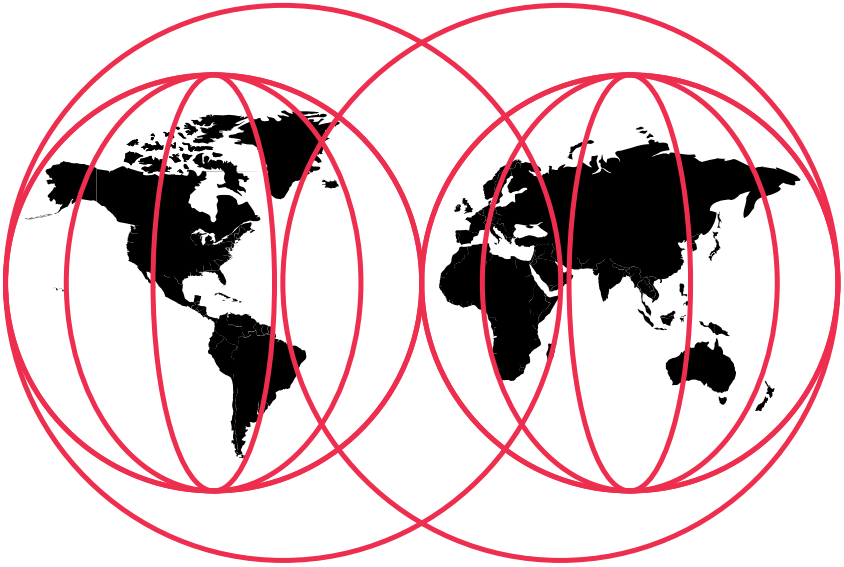
Lotus®

IBM

Creating Customized Solutions with Domino.Doc

Søren Peter Nielsen, Claudio Calazans, Timothy J. Campbell, Pradeep Kumar P.V.



International Technical Support Organization

www.redbooks.ibm.com

SG24-5658-00

SG24-5658-00

International Technical Support Organization

Creating Customized Solutions with Domino.Doc

November 1999

Take Note!
Before using this information and the product it supports, be sure to read the general information in the Special Notices section at the back of this book.

First Edition (November 1999)

This edition applies to Lotus Domino.Doc Release 2.5

Comments may be addressed to: IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
522 South Road
Poughkeepsie, NY 12601-5400

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Preface vii
The team that wrote this redbook vii
Comments welcome viii
1 Domino.Doc overview 1
Domino.Doc defined 1
Domino.Doc capabilities 3
Business needs satisfied 3
Summary 4
2 Domino.Doc architecture 5
Domino.Doc hierarchy and databases 5
Databases 6
File cabinets and server-to-server replication overview 9
Distributed architecture considerations . . . 10
Summary 11
3 Why customize 13
Five main reasons to customize Domino.Doc . . 13
Integration with desktop applications 14
Customized look and feel 14
External application integration 15
Custom processing 15
Integration of third-party technology 15
Customizing Domino.Doc 15
Configuration 16
Document and binder types 16
Design modification 16
Document events handling 18
Domino.Doc API programming 18
Custom agents 19
ODMA 19
Companion products 19
Case study 19
Millennia scenario 20
A note about the examples 22
Summary 24
4 Document and binder types 25
Why document and binder types 25
Defining a document type 25
Designing the document type subform . . . 26
Creating a document type in the library . . . 29
New document type option 30
Applying the document type to a file cabinet . . 43
Another case study document type 45
Defining a binder type 46
Designing the binder type subform 46
Creating a binder type in a library 48
Applying the binder type to a file cabinet . . 49
Rebuilding folders 52
Modifying document and binder subforms . . 54
Deleting document and binder types 54
Summary 56
5 Customizing forms 57
What is a form 57
Forms in Domino.Doc 57
Modifying document-related forms 58
NewDocument 59

Document 60
Customization 62
Modifying search-related forms 66
Notes client search forms 67
Web client search forms 69
Customization 71
Summary 78
6 Customizing views 79
What is a view 79
Domino.Doc views 79
Modifying existing views 80
Adding a tag to new documents for Web users 80
Adding a column to count documents 83
Creating a new view 85
Single category view 90
Summary 92
7 Customizing navigators 93
What navigators are 93
Where navigators are used in Domino.Doc . . . 93
How Web users see Domino.Doc navigators . . 93
Millennia's requirements for modifying navigators 94
Modifying existing navigators 94
Removing unused buttons 94
Inserting Millennia's logo in the existing navigators 96
Modifying navigator colors and buttons . 101
Creating and using a new navigator 106
Using new R5 Design elements instead of navigators 109
Resources 109
Summary 112
8 Document life cycle 113
What the document life cycle is 113
Review and approval cycles 114
Review cycle 115
Approval cycle 120
Review and approval cycles in distributed Domino.Doc setup 126
Summary 126
9 Domino.Doc events 127
Customizing events 127
Events 127
Event handlers 128
Domino.Doc events 128
Domino.Doc event handlers 129
Why use LibEvents and DocEvents 131
Millennia's DocEvents customization . . . 132
Summary 137
10 Domino.Doc API 139
Why use the Domino.Doc API 139
What you need to use the Domino.Doc API . 140
Object model overview 140
Library class 141
Collection classes 145
Rooms collections and Room objects 147
Cabinets collections and Cabinet objects . . 149
Binders collections and Binder objects . . . 151
Documents collections and Document objects 154
Creating documents 154
Getting documents 157
Updating documents 162
Deleting documents 165

Creating an external application with the Domino.Doc API 167
Overview of the application 167
Building the application 171
Summary 184
11 Agents 185
Agents 185
Agents and security 186
Three considerations for creating your agent 187
Web considerations concerning agents . . 189
Domino.Doc agents 190
Millennia's agent requirements 190
Creating Millennia's agents 190
Summary 202
Appendix A Domino.Doc forms . . 203
Forms in the library template 203
Forms in the file cabinet template 204
Appendix B Domino.Doc views . . 207
Library template views 207
File cabinet views 211
Appendix C Domino.Doc navigators 215
Library navigators 215
File cabinet navigators 217
Appendix D Domino.Doc agents . . 227
Library agents 227
File cabinet agents 228
Appendix E Domino.Doc object model 231
DocFileCabinet class 231
DocFileCabinet methods 231
DocFileCabinet properties 231
DocFileCabinet events 232
DocBinderType class 233
DocBinderType methods 233
DocBinderType properties 233
DocBinderType events 233
DocDocumentType class 233
DocDocumentType methods 233
DocDocumentType properties 233
DocDocumentType events 233
DocReplica class 234
DocReplica methods 234
DocReplica properties 234
DocReplica events 234
Retrieval class 234
Retrieval methods 234
Retrieval properties 234
Retrieval events 234
SearchDisplayObject class 235
SearchDisplayObject methods 235
SearchDisplayObject properties 235
SearchDisplayObject events 235
DocDocument class 235
DocDocument methods 235
DocDocument properties 236
DocDocument events 238
Appendix F Domino.Doc API objects and classes 239
Collection classes 239
Room objects 240
Cabinet objects 240
Cabinet objects 241
Binder objects 241
Document objects 242
Security objects 245
User objects 245

Profile objects 246
Field objects 246
Keyword object 247

Appendix G ODMA and the Desktop Enabler 249

Introduction 249
Domino.Doc and ODMA-enabled applications 250
Using the Desktop Enabler 250
Installing the Desktop Enabler 250
Using the Desktop Enabler within applications 252

Appendix H Domino.Doc Storage Manager 261

Storage Manager Connection 262
Storage Server 263

Appendix I Domino.Doc Imaging Client 265

Appendix J Domino.Doc integration with Domino Workflow . 267

Special notices 269

Additional Web material 273

How to get the Web material 274

Related publications 275

International Technical Support Organization publications 275
Other Lotus-related ITSO publications 276
Redbooks on CD-ROMs 277

How to get ITSO Redbooks 279

IBM intranet for employees 279

IBM redbook fax order form 281

Index 283

IBM redbook evaluation 287

Preface

This redbook describes how to create customized solutions with Domino.Doc, providing step-by-step instructions as well as numerous examples. First we describe what Domino.Doc does and give an architectural overview of the product. Then we explain the different ways to customize Domino.Doc, from basic techniques such as Document and Binder Types, through the addition of company logos to forms and navigators, to more complex customization using LotusScript, DocEvents and custom agents, and finally to programming using the API.

All the examples from our case study are available on the IBM Redbooks Web site.

This redbook is written for Domino.Doc solution designers and programmers, customers, IBM Business Partners, and other members of the IBM and Lotus community who need a good technical understanding of how to customize Domino.Doc solutions.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Center at Lotus in Cambridge, Massachusetts, USA.

Søren Peter Nielsen works for the International Technical Support Organization at Lotus Development, Cambridge, Massachusetts. He manages projects that produce redbooks about all areas of Lotus products. Before joining the ITSO in 1998, Søren worked as an IT Architect for IBM Global Services in Denmark, designing solutions for a wide range of industries. Søren is a Certified Lotus Professional at the Principal level in Application Development and System Administration.

Claudio Calazans is a Project Manager and Solution Sales Manager at Tematec Soluções in Rio de Janeiro, RJ, Brasil. Claudio is an expert in vertical corporate solutions that require extensive document management and knowledge management skills in different industry areas, such as government, finance, insurance, and chemicals. He has over eight years of experience in the IT industry and he is a Certified Lotus Principal Professional and Certified Lotus Instructor.

Timothy J. Campbell works in the Network Integration Services department of EDS Systemhouse. His current position is Associate Systems Engineer in Edmonton, Alberta, Canada. Tim's expertise is in designing and implementing complete custom document management solutions using Domino.Doc and various imaging products to solve a variety of business needs, including the incorporation of legal requirements with respect to document retention. He has over four years of experience in the IT industry.

Pradeep Kumar P.V is a Software Engineer from IBM India Limited. He works with the e-business solutions development team, primarily focusing on developing document management, workflow and imaging solutions for customers using IBM EDMSuite products and Domino. He has more than three years of experience in the IT industry. Pradeep is a CLP Notes Principal Application Developer.

A number of people have provided support and guidance. In particular, we would like to thank **Anthony Arcudi**, Product Manager for Domino.Doc. In addition, we would like to thank the following people from Lotus (unless otherwise noted):

- Tom Guyette
- Jason Herrick
- Alison Chandler, ITSO Poughkeepise
- Graphic Services, Lotus North Reading

Comments welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found at the back of this book to the fax number shown on the form.
- Use the online evaluation form found at

  `http://www.redbooks.ibm.com/`
- Send your comments in an Internet note to

  `redbook@us.ibm.com`

Chapter 1
Domino.Doc overview

This chapter provides an overview of Domino.Doc, including what its capabilities are and the general business challenges which Domino.Doc can solve.

Every organization needs to manage and share its documents in different formats among its users and groups. Domino.Doc is the best solution for organizing, managing, and storing all of your critical business documents, and for making them accessible to users inside and outside your organization. Domino.Doc helps streamline multiple business processes, while functioning as a key component of enterprise-wide knowledge management initiatives.

Domino.Doc defined

Domino.Doc is a Domino-based document and content management solution that allows organizations to capture, store, retrieve, and distribute content across the Internet using desktop applications, Web browsers, or any Lotus Notes client. Domino.Doc manages documents throughout their life cycle — from authoring through review, approval, distribution, and archiving.

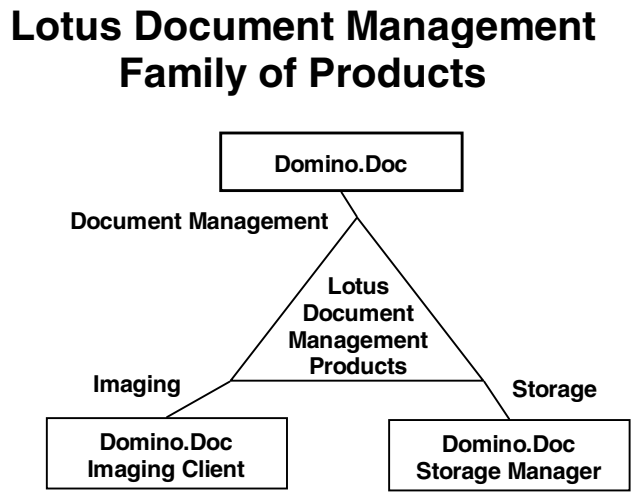
Domino.Doc Desktop Enabler provides tight integration between Open Document Management API (ODMA) compliant desktop applications and Domino.Doc. For applications that support the ODMA industry standard, integrating with Domino.Doc is instant.

The Domino.Doc family of add-on products, Domino.Doc Imaging Client and Domino.Doc Storage Manager, have important roles in imaging and storage management solutions. These products are available separately; they are not included with Domino.Doc.

Domino.Doc Imaging Client provides desktop-based imaging, annotation, and optical character recognition (OCR) capabilities. Using the Imaging Client, you can scan paper documents or import images and store them in a Domino.Doc library.

Domino.Doc Storage Manager provides automated, off-line storage for Domino.Doc. It moves documents from a Domino.Doc library to low-cost storage media like optical jukeboxes or tapes.

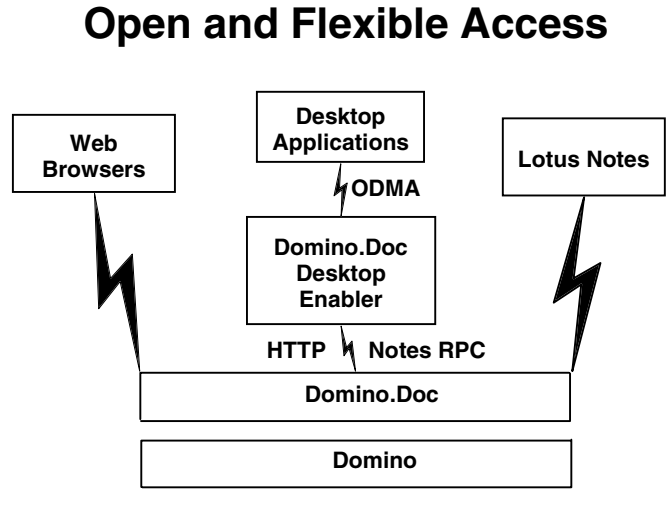
The following figure shows the Lotus Document Management family of products:



Domino.Doc can integrate with Domino Workflow to extend the document management capabilities of Domino.Doc by providing structured and ad hoc workflow processes.

Domino.Doc is a scalable and flexible document management application built on Domino. You can customize it, altering or enhancing the existing functionality to suit your business needs.

The following figure provides a pictorial view of how the different components of Domino.Doc work together on top of Domino's core services:



Domino.Doc capabilities

Among the features provided by Domino.Doc are the following:

- The ability for groups of individuals to collaborate on a set of documents, be they authors, reviewers, or managers.
- The ability to capture, store, retrieve, and distribute documents across virtual communities, from Notes clients, Web browsers, or direct from ODMA-enabled Windows applications.
- An enterprise library for storing documents.
- A familiar metaphor of library, file rooms, file cabinets, folders, and documents.
- Management of documents throughout their life cycle — from authoring through review, approval, distribution, and archiving.
- Support for all the key functionality users need, like version control, check-in and check-out, document profiling, and search and retrieve, all on top of a robust, secure Domino environment.
- Compliance with the ODMA industry standard for easy storage by users. You can work in the applications you are familiar with. From within an ODMA application, you are able to save documents into Domino.Doc in a direct and simple manner.
- Full intranet and Internet accessibility.
- A program built on Domino architecture for scalability replication.
- The ability to integrate with Domino.Doc Storage Manager for cost-effective document storage.
- The ability to integrate with Domino Workflow to leverage its powerful workflow capabilities.

Business needs satisfied

Domino.Doc satisfies the following business needs:

- Centralized document repository

  Domino.Doc enables you to have a common place to store electronic documents, instead of storing them on various file servers and desktop machines located around the company. This provides a consistent and logical way to store documents.

- Review and approval cycle management

  Domino.Doc enables you to define review and approval cycles by document type so that all documents of a particular type will pass

through a set of reviewers and approvers before they are available to end users. This helps the appropriate managers to have full control over documents.

- Document searching

  The document profile feature enforces the capture of key document search terms so that you can easily search and retrieve documents. You provide necessary and important profile information for each document; later you can search for a document based on this profile information. This reduces the time spent on searching and retrieving documents.

- Data integrity

  Version control and content update control, using document check-in and check-out features, ensure data integrity. You can check out documents for changes, and others can still browse and read them but not make changes which will overwrite yours. In this way, only the correct and latest information will be available.

- Document security

  Document security is a major concern in many organizations. Domino.Doc, being built on Domino, ensures high security for documents. Domino.Doc authenticates Web users before granting them access to the data. You can have security at the file cabinet, binder, and document level. This way you are able to protect your critical business information from unauthorized access.

Summary

In this chapter, we provided an overview of Domino.Doc, its capabilities, and how Domino.Doc can satisfy your business needs.

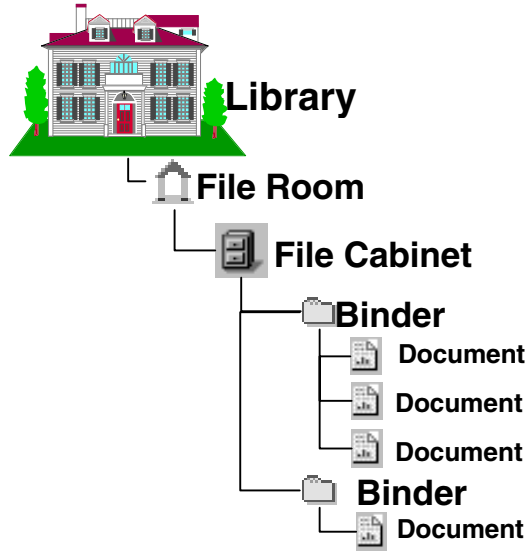
Chapter 2
Domino.Doc architecture

This chapter outlines the general architecture of Domino.Doc. This chapter is divided into two sections: the first section reviews the system requirements and describes each type of database used by Domino.Doc; the second section discusses library replication, as well as some considerations to take into account when working with a distributed library.

Domino.Doc hierarchy and databases

In this section we discuss the hierarchy Domino.Doc uses to organize documents, and the templates and databases that form the nuts and bolts of Domino.Doc.

The following figure illustrates the metaphor on which the Domino.Doc file hierarchy is organized. It is followed by a brief description of each element.



Note There is an additional layer above library as of Domino.Doc 2.5 that best can be described as *Site*. That is a meta-layer for administration of multiple libraries from one central place. Libraries are intended to be completely independent except at the administrative level.

Library

The *library* is your entry point to Domino.Doc. The library contains the "main view," which you use to navigate through the entire library, to perform searches, and to perform a number of administrative tasks.

File Room

The *file room* is a logical construct used to organize file cabinets. When you create a file cabinet, you will have the option of choosing to which room(s) you will add the new file cabinet. You will have a choice whether to put it in an existing file room, or to create a new file room.

File Cabinet

Domino.Doc uses *file cabinets* to organize both binders and documents. File cabinets are made up of Domino databases. The file cabinet may be used for accessing the document information as well as viewing, storing, retrieving, managing, and distributing all the documents in the library.

Binder

A *binder* is a container within a file cabinet that is used to store and organize your documents.

Document

A *document* can be defined as any bit of information that you wish to store in the Domino.Doc library. The document may have originated in either paper form (and been scanned) or in electronic form (such as a Word Pro or Freelance Graphics file).

Databases

This section outlines the different types of Domino databases used by Domino.Doc, as well as describing the function of each database. The different types of databases are:

- Domino.Doc Site Admin database (one per server)
- Domino.Doc Transactions database (one per server)
- Library database (one per library, or one per replica of a library)
- Configuration databases (one per library, or one per replica of a library)
- Log databases (one per library, or one per replica of a library)
- File Cabinet databases (one Binder database per File Cabinet, and one or more Document databases per File Cabinet)
- Library design template (either one per library, or one shared for all libraries)

- File Cabinet design template (either one per library, or one shared for all libraries)

Domino.Doc Site Admin database (ddadmin.nsf)

The Domino.Doc Site Admin database is where you create and maintain Domino.Doc libraries. When creating a master library, you may choose between a "Quick and Easy" setup, which uses the most common settings, or an "Advanced" setup which allows you to modify the settings. This database can also be used to create a replica library.

Domino.Doc Transactions database (ddmtrans.nsf)

This database is used, in conjunction with the Transaction Manager (a Domino add-in task) to replicate any changes to documents, binders, and file cabinets, as well as to process check-out requests.

Library database (*xyz*Lib.nsf)

This is the database that your users will normally open to begin their Domino.Doc session. There is one Library database per Library (or one Library database per replica of a library). This database contains information, such as pointers, to access each File Cabinet.

Configuration database (*xyz*Cfg.nsf)

The Configuration Database contains the definition (not design) of all the binder and document types. The File Cabinet Databases use this information to select the correct subform for the type selected by the user when creating a new binder or document.

Caution Do not modify a Configuration Database. Domino.Doc will automatically do this when you change the definitions from within the Library database.

Log database (*xyz*Log.nsf)

This database is used to inform you of errors and activity in the Library. There is one Log Database per Library (or per replica of a Library).

File Cabinet databases (*abc-123*.nsf)

The File Cabinet databases are the core of the system. There are two types of File Cabinet databases: Binder databases and Document databases.

There is one Binder Database per File Cabinet. It contains a list of all binders in that File Cabinet, as well as the information required for Domino.Doc to open the correct Document Database when a binder is selected.

The Document Databases contain the Documents entered by the users. Depending on the options you selected during the creation of the File Cabinet, there may be multiple Document Databases per File Cabinet.

Library design template (domdoc.ntf by default)

This is the template for customizing the design of the Library database. By choosing the "Quick and Easy" setup option in the Domino.Doc Site Admin database (when creating a Library), your Library database will be based on the domdoc.ntf template. By choosing the "Advanced" setup option, you have the choice of accepting domdoc.ntf as the template, or you may enter the name of another template. To use a different template, you must do the following:

- Create a copy of the domdoc.ntf file, and rename it. Be sure to keep the .ntf extension so that Domino recognizes it as a template, and be sure that it is in the data directory so that Domino can find it.
- Use the "Advanced" setup option in the Domino.Doc Site Admin database to create the Library.
- Enter the name of your template in the field "Library Design Template" (replacing the default text, "domdoc.ntf").

  Repeat these steps for each Library that will have a unique Library database design. Be sure to give each copy of the template a unique name, otherwise you will overwrite your other templates. Instructions for modifying this template are discussed later in this book.

Tip It is good practice to make a backup of the original domdoc.ntf file before modifying it!

File Cabinet design template (filecab.ntf by default)

This is the template for customizing the design of the File Cabinet database. By choosing the "Quick and Easy" setup option in the Domino.Doc Site Admin database (when creating a Library), your File Cabinets in the Library by default will be based on the filecab.ntf template. By choosing the "Advanced" setup option, you have the choice of accepting filecab.ntf as the template, or you may enter the name of another template. To use a different template, you must do the following:

- Create a copy of the filecab.ntf file, and rename it (be sure to keep the .ntf extension so that Domino recognizes it as a template, and be sure that it is in the data directory so that Domino can find it).
- Use the "Advanced" setup option in the Domino.Doc Site Admin database to create the Library.
- Enter the name of your template in the field "File Cabinet Design Template" (replacing the default text, "filecab.ntf").

  Repeat these steps for each Library that will have a unique File Cabinet database design. Be sure to give each copy of the template a unique name, otherwise you will overwrite your other templates. Instructions for modifying this template are discussed later in this book.

Note It is possible to specify a design template at File Cabinet creation, and different cabinets in a single Library can be based on different templates.

Tip It is good practice to make a backup of the original filecab.ntf file prior to modifying it!

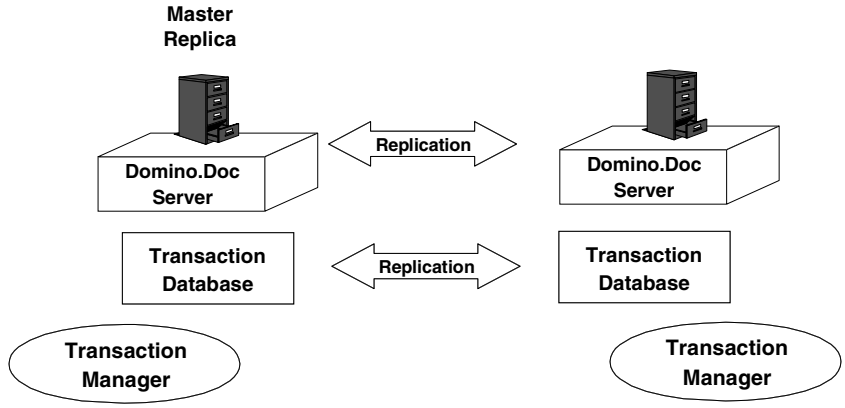
File cabinets and server-to-server replication overview

The Domino.Doc server-to-server replication feature lets you keep multiple copies, called replicas, of a file cabinet on multiple servers. Through replication, all of the replicas become essentially identical over time.

For example, users in one office can make changes to a replica on their servers at the same time that users in another office make changes to a replica of the same file cabinet on their server. When the servers replicate, each replica is updated with the information from the replica on the other server.

In order to avoid replication conflicts caused by changes made to the same document or simultaneous check-out requests by users who are working on different servers, during the installation process the installation program copies a Domino server add-in task called Transaction Manager to the program directory. This task is responsible for managing documents ("events") posted into the Transactions Database.

See the figure below for an example of Domino.Doc in a distributed Environment.



There is a designated *master replica* server. We have a master library replica and a master replica for each file cabinet. However, the master replicas for the different file cabinets do not have to be on the same server. Certain types of operations like file cabinet creation/deletion, update/lock requests are restricted to the master replica server. To communicate and coordinate these operations there is a Transactions database and a Transaction Manager server add-in task on each Domino.Doc server. The Transaction database contains the data for the transactions themselves. The transactions are transmitted between Domino.Doc servers through replication. When a server needs to send a Domino.Doc transaction to another server, it opens the local copy of the Transaction database and creates the necessary transaction. When replication occurs, the transaction arrives in the necessary Transaction database replica on the destination server. The Transaction Manager monitors the Transaction database and processes the transactions as they arrive.

Distributed architecture considerations

This book is not intended to describe how Domino.Doc replication takes place or how to configure two Domino.Doc servers to replicate each other. If you are using Domino.Doc in a distributed environment there are some important things you should consider before customizing your applications as they may affect the document check-in and check-out process.

Before starting to customize Domino.Doc in a distributed environment, you should check the following:

- Available disk space on the destination server
- Users' access rights
- Replication and mail routing topology and intervals
- Whether or not the replica of the file cabinet resides on a clustered server
- Location of the home library of this cabinet

Available disk space on the destination server

Replication can only occur at the library or file cabinet level. This means that you have no control over the replication of individual binders or documents.

In order to preserve disk space you should try to fill the cabinet with documents that the users will really need to access on the destination server; other documents should be located in cabinets that are not replicated to the users' server.

Access to server

If your customization forces users to access cabinets that are located in different Domino.Doc servers, be sure that they have the appropriate access rights to those servers.

Replication and mail routing topology and intervals

You should check the replication and mail routing topology in order to avoid users receiving workflow messages with links that point to documents which actually do not exist in the file cabinet. This would give them error messages, because the servers have not replicated yet.

Using Domino.Doc on clustered servers

If your Domino severs are clustered and the Domino.Doc databases are going to be part of the clustered databases, do not forget to include the Transactions Database on your clustered database list.

Home library location

Keep in mind that, if the users are not able to check out a document directly from their file cabinet replicas, you will need to customize your application so that your users will not have to deal with check-out rejections or check-out conflicts which they do not understand.

Note You may want to replicate Domino.Doc across multiple domains if your company works closely with, and needs to communicate with, another company.

Working with multiple domains requires additional administrative work.

You must enable Directory Assistance, enable the Administration Process, and cross-certify all users on all servers where Domino.Doc will be enabled. However, having Domino.Doc replicas across domains does not actually affect your customization in any other way.

Summary

In this chapter, we discussed the overall architecture of Domino.Doc. We reviewed the different templates and databases in the Domino.Doc hierarchy. Finally, we discussed Domino.Doc replication and the considerations for working with a distributed library.

Chapter 3
Why customize

This chapter introduces the topic of customizing Domino.Doc. You will learn why you might want to customize Domino.Doc when building a document management solution. You will also learn the methods available to you for customization.

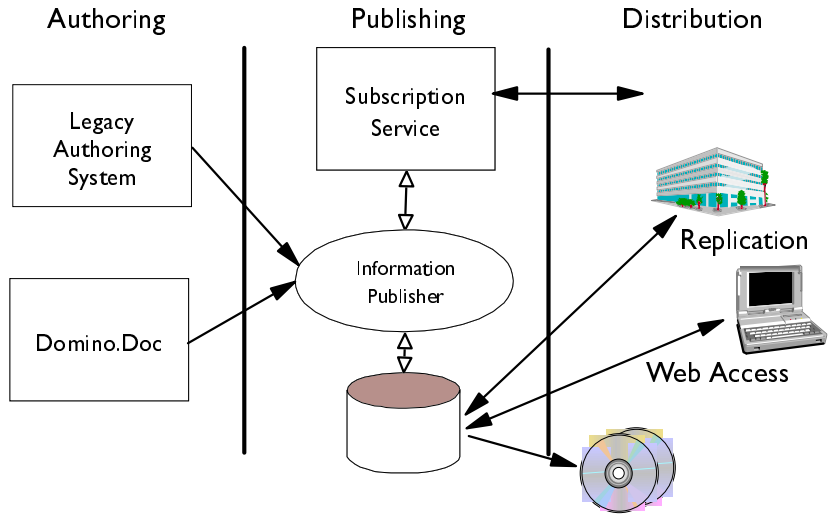
The final section of this chapter will introduce to you our sample customer, Millennia Space Travel Corporation, and their requirements for a complete document management solution. This scenario will be used throughout the book to demonstrate how each customization relates to others to build a complete document management solution.

Five main reasons to customize Domino.Doc

Although there are an unlimited number of reasons why you might choose to customize Domino.Doc solutions, there are five key reasons that are often seen when customers define their requirements. These five main reasons are as follows:

- Integration with desktop applications
- Customized look and feel
- External application integration
- Custom processing
- Integration of third-party technology

The following figure shows a typical document management environment, which would require customizing your Domino.Doc solution.



In the above figure, Domino.Doc is shown as part of the entire solution. An Information Publisher takes documents from both Domino.Doc and a legacy system, and then publishes them to the Internet, to media storage, and to an off-site database. The Information Publisher also publishes the information to an e-mail based subscription service. This is an example of the typical requirements for a complete customized document management solution.

The five key customization reasons are described in detail in the following sections.

Integration with desktop applications

Domino.Doc can be integrated with most common desktop applications with the Open Document Management API (ODMA) standard by using the Domino.Doc Desktop Enabler. Other applications, which do not support the ODMA standard, may still need to be integrated with Domino.Doc. In this case, you may need to customize your Domino.Doc solution to include this integration.

Customized look and feel

Customizing the look and feel of Domino.Doc includes adding company logos, changing color schemes, modifying terminology, and enhancing the user interface.

External application integration

Integrating external applications refers to customizing Domino.Doc with other Domino applications, other database systems, other document management software, and any other external applications.

When using Domino.Doc to replace a legacy document management system or other document storage application, such as a Lotus Notes Document Imaging (LN:DI) solution, you will have to customize Domino.Doc in order to migrate data from the old system.

Custom processing

You can add to or change system functionality, such as the handling, interception, or redirection of documents, by customizing your Domino.Doc solution.

An example of this is an automatic notification to sales people in an organization when a marketing presentation is approved for publication.

Integration of third-party technology

Many additional technologies can be integrated with Domino.Doc, such as third-party Internet applications, scanning solutions, or publishing tools. This is another reason to customize Domino.Doc.

Customizing Domino.Doc

There are many ways to customize a Domino.Doc environment. The methods used to customize the system at your site depend on your particular requirements.

The following are the major methods by which you can customize your Domino.Doc installation:

- Configuration
- Document and binder types
- Design modification
- Document events handling
- Domino.Doc API programming
- Custom agents
- ODMA
- Companion products

These methods are described in the following sections.

Configuration

You can achieve some level of customization when you configure your Domino.Doc components, by selecting or deselecting some of the available options.

The components you can configure include:

- Users and privileges
- File cabinets
- Binders and document types
- Full text indexing
- Agents
- Database replication
- Life cycle management
- Error logging
- Transactions

Document and binder types

Domino.Doc comes with a set of document and binder types. In addition to these, you can create your own document and binder types to customize your application. This associates a set of attributes with the documents or binders created with that type. Document and binder types are created based on subform selection. The fields and formulas in the subform will be the attributes in your document and binder profile.

Add new subforms in the file cabinet template with the necessary fields and formulas to meet your requirements, and create new document and binder types based on these subforms.

Design modification

Domino design elements such as forms, views, subforms, and navigators provide a major tool for customizing Domino.Doc solutions. You can modify the existing design elements that come with Domino.Doc, or create your own to suit your requirements.

The following sections describe some customizations you can achieve by modifying these design elements.

Modifying forms

Possible modifications to forms include the following:

- Reorganize the form layout, for example, to fit the most important information on a single screen. This way users don't have to scroll down to see it all.
- Include new fields on a form, for example, add fields to display the creator of the document and the creation date.
- Change the form's style or background to conform to established company standards. For example, you can include the company name or logo as watermarks in the form background.
- Insert the company logo.
- Insert new graphical elements, for example, add a graphic to indicate the application type: discussion, legal, or marketing.
- Rearrange the action buttons, or change their wording or icons.

Modifying subforms

The possible modifications you can make to subforms include the following:

- Include new fields for your requirements.
- Insert company logo.
- Place new graphic elements.

Modifying views

The possible modifications you can make to views include the following:

- Include new columns, or change the existing columns.
- Modify the categorization.
- Modify the selection criteria.
- Rearrange the action buttons, or change their wording or icons.
- Use different colors to change the look and feel.

Modifying navigators

The following are some of the modifications you can make to navigators:

- Insert company logo.
- Place new graphic elements.
- Create image maps.
- Hide administrative tasks from ordinary users.
- Remove buttons which are not relevant for your application.

Document events handling

Use the Domino.Doc event model to customize your application. Each Domino.Doc event has a corresponding event handler; you can program these event handlers for your custom processing. Program the *Query* and *Post* event handlers for the following events:

- Add/Create
- Delete
- Review
- Approval
- Check in/Check out

In addition, you can program the event handlers for replicas and search display. The event handlers can also be used for:

- Validation checking
- Setting default values
- Triggering other custom processing
- Bypassing regular workflow
- Running agents

Domino.Doc API programming

Using Domino.Doc API programming for customizing your solution is a good way to add a great deal of functionality to your system.

Use API programming for:

- Custom processing, like changing field values of documents already created
- Creating a custom user interface for your applications
- Bypassing the Domino.Doc user interface for background processes like importing documents
- Accessing the Domino.Doc library from other desktop applications and import/export programs
- Generating reports

The Domino.Doc API provides a set of classes which you can use to access the data inside the library.

Custom agents

Create Domino agents to achieve some degree of customized automation. Agents are useful for the following:

- Archiving documents

  Archive documents automatically based on specified criteria.

- Notifying users

  Notify Domino.Doc users by e-mail based on specified document status.

- Gathering statistics

  Collect data about users, documents, and life cycle events. This data can be useful for analyzing and reporting.

- Changing ACL

  Revise the user list for a document or binder, add or delete a user, or change the entire list to move a set of documents to a different group altogether.

- Moving documents

  Move documents to selected binders, or move documents temporarily to default binders under certain conditions.

ODMA

Use the ODMA standard to create your own applications, or transfer existing applications to ODMA-aware applications and connect with Domino.Doc to access the data.

Companion products

Use companion products to customize your overall application. For example, use scanners, big screen monitors, and other products to get more customization for your organization.

Case study

This section describes the fictitious company *Millennia Space Travel Corporation* and its requirements for a document management system. We will use this company for the examples throughout this book.

Millennia scenario

Millennia Space Travel Corporation is a small company with 3000 employees distributed on six sites: Earth, Mars, Venus, Jupiter, Pluto, and Serydd, in the Cygnus A galaxy.

Millennia's specialty is bringing people to and from other planets across the Milky Way and other galaxies.

Millennia management is unhappy with the way they are currently storing and managing their documents. They would like to implement one solution that allows their employees to access and collaborate on document content across the organization. This solution must be compatible with the common Internet standards, such as Hyper Text Transfer Protocol, for access from Web browsers.

The marketing team manager would also like to be able to make marketing presentations available to users over the Internet, providing them with a new way to find travel destinations that match their lifestyle.

Millennia requirements

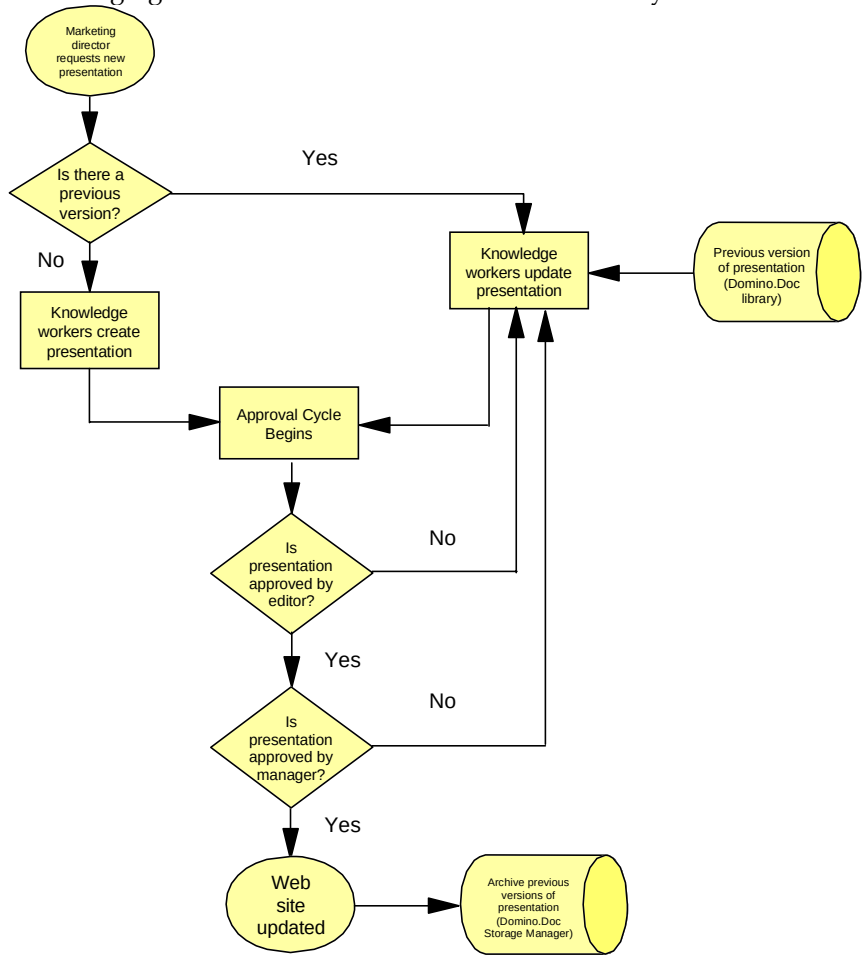
After one decisive meeting the Millenia group was able to define their requirements as follows:

- Have a user-friendly application which stores all Millennia's documents.
- Give users the capability to manage the review and approval process for marketing presentations before they are published to the Internet.
- Archive old presentations when new versions are created.
- Publish planets' policies and laws in their original appearance.
- Enable users inside the company to find documents based on the document category.
- Notify users whenever there is new sales information available that meets their personal interests.

Note In a real-world situation there would most likely be more requirements. The sample requirements provided here demonstrate the general principles of customizing a Domino.Doc solution.

Millennia's marketing document flow

The following figure illustrates Millennia's document life cycle:



Millennia's reasons for using Domino.Doc

The key factors in Millennia's decision to use a Domino.Doc solution are the following:

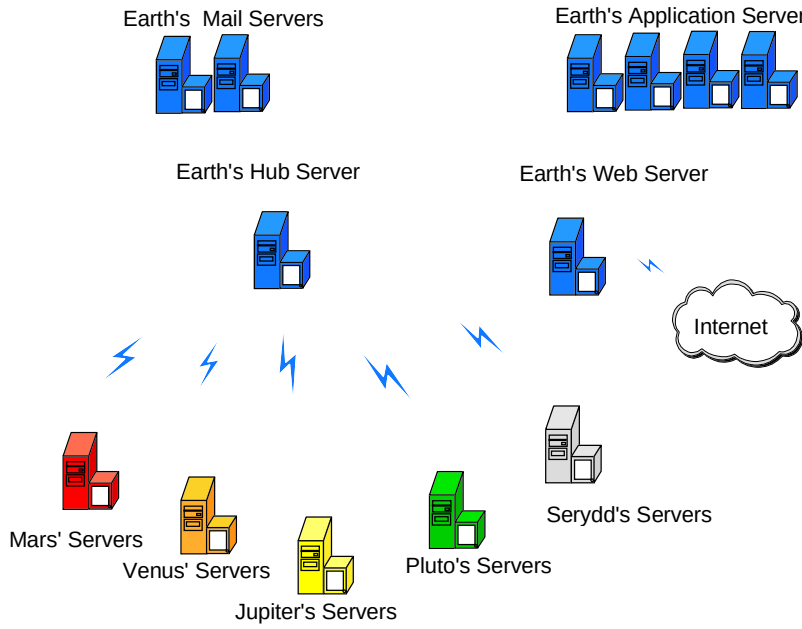
- Domino.Doc is totally integrated with Domino, which is a platform that they are currently using.
- An enterprise-wide Domino.Doc deployment would offer scalability, reliability, and manageability, and be cost-effective.
- Millenia will not need to start from scratch in training users, because they already use Domino.

- Domino.Doc is accessible to Web browser users.
- Millenia can customize Domino.Doc in order to meet their requirements for document and knowledge management.
- Complementary technologies are available.

Millennia's actual Domino structure

Millennia has seven Domino servers on Earth and two on each of the other planets. These servers are replicating each other, and each site has one server that replicates with Earth's hub server using a high speed satellite link.

The following figure shows Millennia's Domino servers:



A note about the examples

Throughout the rest of the chapters in this book we will show examples of how we modified Domino.Doc for our fictitious company, Millennia Space Travel Corporation.

If you want to try out the examples as you go along, you should note the following:

- The steps in the examples are described using Domino Designer R5.0. Users of Designer for Domino R4.6 will sometimes experience slightly differently wording or different window layout in the Designer.

- We only describe the parts of the Domino design elements that are relevant to the examples. We assume some knowledge of Domino programming concepts such as forms, views, agents and so on. If you want to learn more about Domino programming refer to *Lotus Domino R5.0: A Developer's Handbook*, IBM form number SG24-5331, Lotus part number CT6HPIE.
- You need a Domino.Doc installation to see the results of the modifications, and the user ID used for the modifications must have Domino.Doc Administrator access. We will not give a detailed description of how to install Domino.Doc. However, if you are making a test installation of Domino.Doc you can install it using all the default values, and the user ID you are using will automatically get Domino.Doc Administrator access. For the library and file cabinet setup here is the information you need to follow our examples:
  - You create your libray from the Domino.Doc Site Admin database. You must use folders in binders, not the BinderTOC control that has been added in Domino.Doc 2.5. If you select Quick and Easy Setup your file cabinets will be set up to use the BinderTOC control. You must use Advanced Setup when you create your library. You can use the default values for the rest of the choices under Advanced Setup.
  - Once you have created your library, you must restart the Domino server and then open your newly created library to create two file cabinets. They should be named *Presentations* and *Interplanetary Policies*. You can originally create them using the default settings. We will walk you through the necessary set up changes in the appropiate examples.
- If you want to save some time keying in the examples, you can download the Domino.Doc library and file cabinet templates with the modifications we have added to them from the IBM Redbooks Web site. Also available for download is a Web site database and Lotus Approach database we use in Chapter 11, "Domino.Doc agents," to show how you can access relational data from Domino and the Visual Basic source files for the external application we create in Chapter 10, "Domino.Doc API."

  See the appendix, "Additional Web material," at the end of the book for a list of additional Web material for this book and instructions on how to get the material.

Summary

In this chapter we described a number of reasons why you might want to customize a Domino.Doc implementation. The methods available to customize the product were discussed. The fictitious company Millennia Space Travel Corporation was introduced, as their requirements will be satisfied through the examples in this book.

Chapter 4
Document and binder types

In this chapter, we discuss document and binder types, how to create new document and binder types for Millennia Space Travel Corporation and how to apply them to the file cabinets.

Why document and binder types

All documents and binders created in Domino.Doc library should have some identifying profile information so that later they can be searched and retrieved. This profile information may be different for different kinds of documents. For example, a proposal document might have information like Company Name, Project Budget, Project Schedule, and Responsible Group. A correspondence document might have information like To, From, Subject and Date. A document or binder type defines the profile information for the documents or binders created with that type.

Document and binder types are created based on subforms. The documents and binders use the fields defined in the subform of the associated type as profile fields. Every document and binder in Domino.Doc should have an associated type. If no type is specified, the document or binder will go to No Type by default.

Defining a document type

A document type associates a set of profile attributes with the documents created with that type. You can make any or all of these attributes required, thus preventing the user from saving a new document of this type without entering necessary profile information. Domino.Doc comes with a set of document types. Additionally, you can create your own document types.

To create a new document type, complete the following tasks:

- Design the document type subform in the file cabinet template.
- Create a new document type in the library.
- Apply the document type to a file cabinet.

The detailed instructions for each task are presented in this section.

Designing the document type subform

The first step in creating a new document type is to add a subform to the file cabinet template. The fields and formulas in the subform will be the attributes in the document profile.

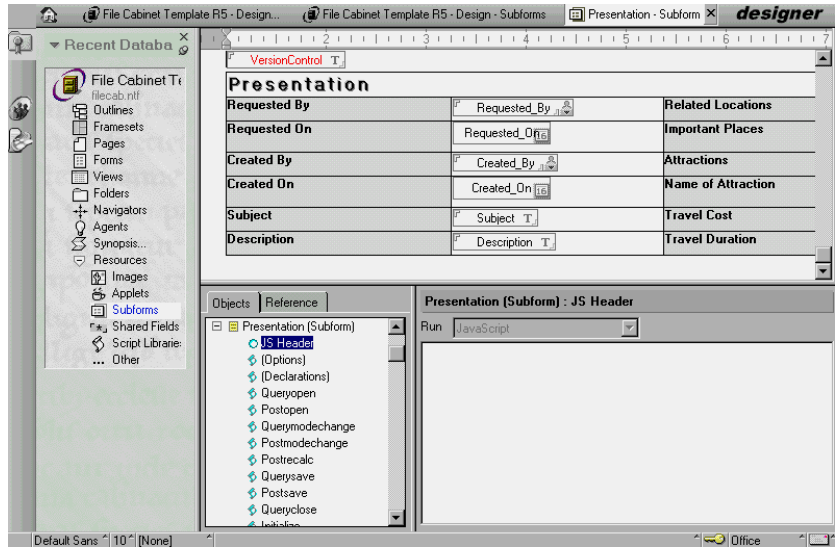
We will design a subform called "Presentation" for Millennia Space Travel Corporation.

To design a subform, perform the following steps:

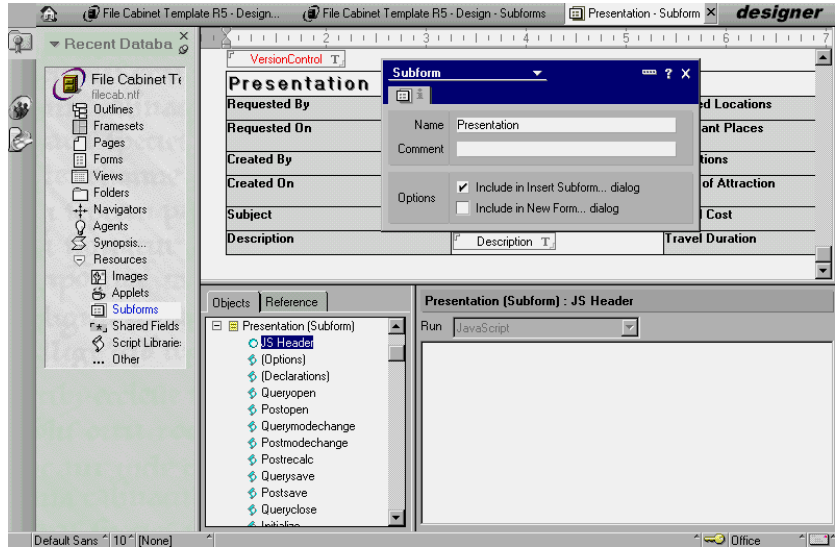
1. Open the File Cabinet Template (filecab.ntf) in Domino Designer and choose Create - Design - Subform.
2. We will now add fields to the subform. You do this by choosing Create - Field. This places a new field on the subform at the cursor location, and open the Properties box for the field where you specify field name, type, and so on.

   Add the following fields, with the field attributes given, to the subform:

Field name	*Type*	*Other options that need to be changed*
Requested_By	Names, Editable	
Requested_On	Date/Time, Editable	
Created_By	Names, Editable	
Created_On	Date/Time, Editable	
Subject	Text, Editable	
Description	Text, Editable	
Locations	Text, Editable	Allow multiple values
Places	Text, Editable	Allow multiple values
Attractions	Dialog list, Editable	Choices - Beach, Restaurant, Museum, Park, Art Gallery
AttractionName	Text, Editable	
Cost	Text, Editable	
Duration	Text, Editable	
VersionControl	Text, Editable	Hidden (This field is used for comparing the document version in DocEvents)



3. Select Design - Subform Properties. The Properties dialog box opens.
4. In the Name field, enter Presentation (The subform name will be the name of your new document type).
5. Ensure that the option "Include in Insert Subform....dialog" is checked.



6. Save and close the subform.

7. To make the new subform available for the users the design or the production databases need to be refreshed from the template we just modified. You can do this manually for each database using File - Database -Refresh Design. Or you can do this for all databases by forcing the design task to run on the server. You this by typing the following command on the server console:

   ```
   load design
   ```

Note Be careful if setting validation formulas for your fields. If a validation formula is incorrect, the user may not be able to save a new document based on this subform.

Field type considerations

If you will be using Domino.Doc with ODMA-enabled applications, you should plan your document type fields carefully to ensure that your ODMA users will see the information intended and can enter the expected attribute values.

The text you enter on the subform to describe a field will not be seen in the ODMA document profile field. Instead, the ODMA user will see the field name next to the input box. Therefore, we recommend naming the field the same as the field descriptive text that will be seen in the browser and in Domino. For example, if the descriptive text is Company Name (to indicate that you want the user to enter the company's name in the input box), name the field Company_Name (underbars are converted to spaces). When you see the document type attribute in an ODMA application, you will see the same information you would in the browser or in Domino for the same attribute.

You should consider the following when creating your document type fields:

- Editable keywords with computed keyword formulas are supported.
- Computed fields are not seen.
- All fields except keyword fields are treated as simple text. Number and time fields are validated only when edited in a browser or Domino.
- Keyword fields always appear as a simple, single selection list box even if check box, radio button, multi-values, or values not in the list are specified.
  - If a keyword formula and a default are supplied, the default will be displayed.
  - If no default is specified, the first keyword in the list will be displayed.

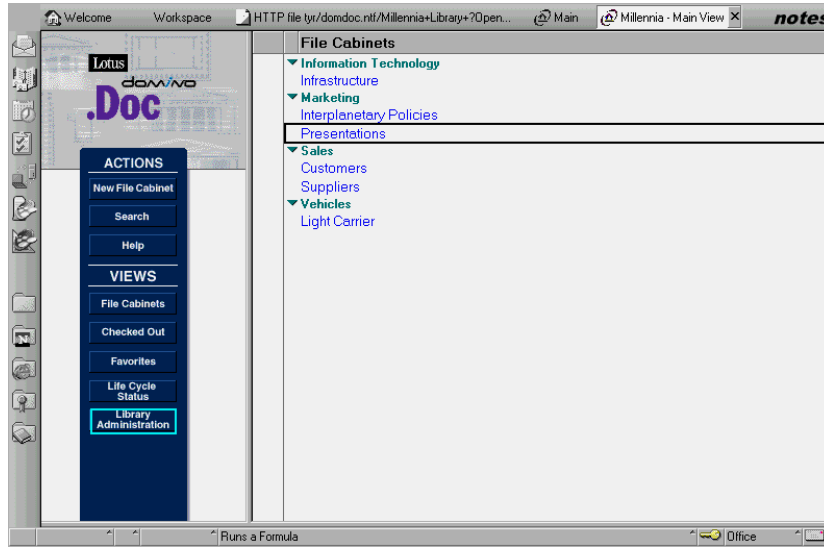
If a field uses the @DBColumn or the @DBLookup function, the formula must evaluate to a non-null value. Otherwise, the user will get “Incorrect data from server” error.

Creating a document type in the library

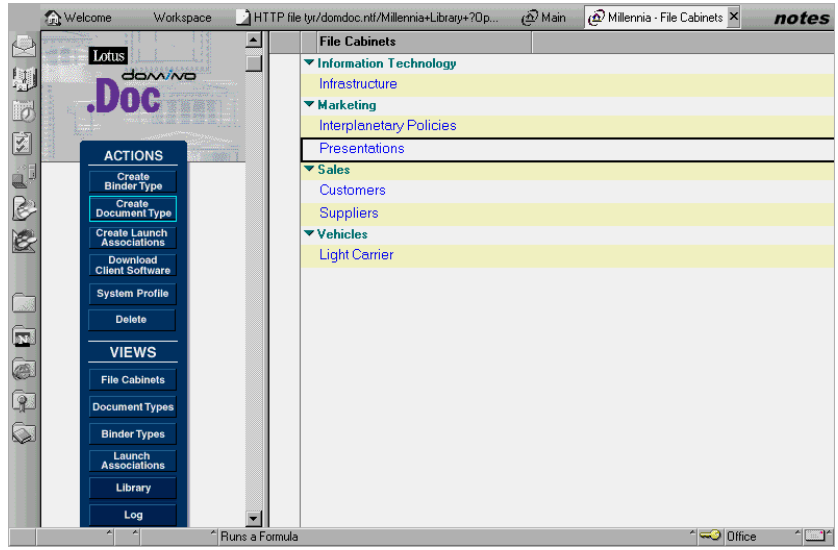
After creating the document type subform in the file cabinet template, add the new document type to the library.

To add the new document type to the library, perform the following steps:

1. Open the Millennia library from a Notes Client by double-clicking the database icon.
2. Click Library Administration on the main navigator.



3. Click Create Document Type. The new document type form is displayed.



4. Complete the sections of the new document type. There are a lot of options that we will cover after the end of this sequence.
5. Save and close the new document type form.

New document type option

Here we walk through the details of step 4 above, where we specify a new document type for Millennia Travel Agency.

- Enter the document type name as Presentation (the name of the document type subform you created).
- Enter the file cabinet template name as filecab.ntf. The default value of this field will be filecab.ntf. You can change this value to the template name you are using for your application.
- Field Attributes

  In this section, specify the field attributes you want to make required, so that users will not be able to save a document of this type without entering values for these fields.

  Select Requested By, Created By, and Locations as required fields for the Millennia application.
- Version Options

  In this section, specify how Domino.Doc should handle documents of this type when they are checked in. Specify the default action for checking in a version and the default action for checking in a draft.

The version parameters are:

What is the default action for checking in a version?

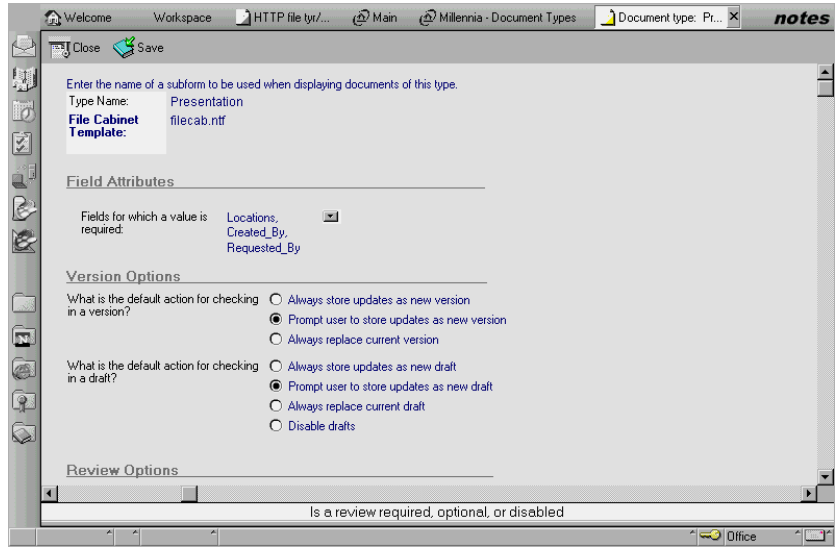
- Select “Always store updates as new version” if you want to automatically create a new version from an edited version and maintain the original.
- Select “Prompt user to store updates as new version” if you want to give the user the option of either creating a new version that maintains the original, or creating a new version that replaces the original.
- Select “Always replace current version” if you want to always replace the current version with the updated version.

Select “Prompt user to store updates as new version” for the Millennia application.

What is the default action for checking in a draft?

- Select “Always store updates as new draft” if you want to automatically create a new draft from an edited draft and maintain the original.
- Select “Prompt user to store updates as new draft” if you want to give the user the option of either creating a new draft that maintains the original, or creating a new draft that replaces the original.
- Select “Always replace current draft” if you want to always replace the current draft with the updated draft.
- Select “Disable drafts” if you want to check in all documents of this type as versions only. This option disables all review and approval capabilities.

Select “Prompt user to store updates as new draft” for the Millennia application.



Review Options

Specify default review parameters for the documents of this type in this section, provided that you did not select the “Disable draft” version option.

The review parameters are:

`When should documents be reviewed?`

- Select “Never” if documents of this type do not need to be reviewed by others.
- Select “Optional” to allow the draft editor to decide whether or not to initiate a review cycle.
- Select “Always” to require reviews of documents of this type. In this case, the user cannot check in the document as a version until the review cycle is complete.

Select“Always” for the Millennia application.

`Default reviewers`

- Select “All draft editors” to make all draft editors reviewers.
- Select “All document managers” to make all document managers reviewers.
- Select “Specify individual reviewers” to make some specific individuals reviewers. In this case you should specify the names of the reviewers.

Select “Specify individual reviewers” for the Millennia application and specify Maria Fernandas/Millennia@Millennia and Dee Jonson/Millennia@Millennia as the reviewers.

`Routing type`

- Select “Serial” if you want each draft editor to review the same copy of the document. The reviewers will review the document in the order specified in the list.
- Select “Parallel” if you want each draft editor to review a separate copy of the document. The reviewers will review the document at the same time.

Select “Serial” for the Millennia application.

`Time limit`

- Select “Send reviewer a reminder after the time limit expires” if you want to notify the reviewer after a specified time.
- Select “Send originator a reminder after the time limit expires” if you want to notify the originator after a specified time.

Select “Send reviewer a reminder after the time limit expires” for the Millennia application and specify the number of days for notification as three.

`When should the originator be notified?`

- Select “After final reviewer” if you want to notify the originator only after the final reviewer.
- Select “After each reviewer” if you want to notify the originator after each reviewer.

Select “After final reviewer” for the Millennia application

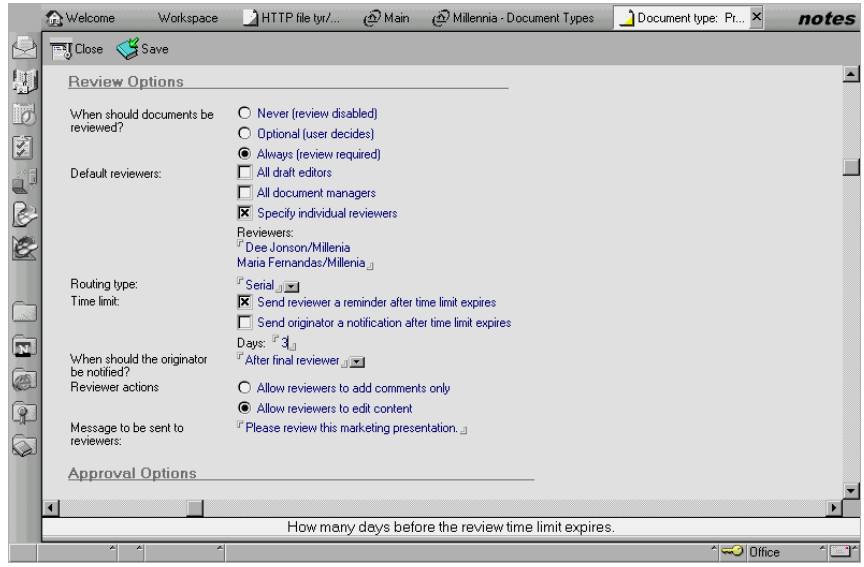
`Reviewer actions`

- Select “Allow reviewers to add comments only” if you want to allow the reviewers only to add their comments and not to modify the content of the documents.
- Select “Allow reviewers to edit content” if you want to allow the reviewers to modify the content of the documents.

Select “Allow reviewers to edit content” for the Millennia application.

`Message to be sent to reviewers`

Enter the message you want to sent to reviewers. This field is optional. Enter “Please review this marketing presentation” in this field.



Approval Options

In this section, specify default approval parameters for the documents of this type, provided that you did not select the “Disable draft” version option.

The approval parameters are:

`When should documents be approved?`

- Select “Never” if documents of this type do not need to be approved by others.
- Select “Optional” to allow the draft editor to decide whether or not to initiate an approval cycle.
- Select “Always” to require approval of documents of this type. In this case, a user cannot check in the document as a version until the approval cycle is complete.

Select “Always” for the Millennia application.

`Should users be allowed to modify the list of approvers?`

- Select “Yes” if you want to allow the users to modify the approvers list.
- Select “No” if you do not want to allow the users to modify the approvers list.

Select “No” for the Millennia application.

`Enter the default approvers`

- Select “All draft editors” to make all draft editors approvers.
- Select “All document managers” to make all document managers approvers.
- Select “Specify individual approvers” to make some specific individuals approvers. In this case you should specify the names of the approvers.

Select “Specify individual approvers” for the Millennia application and specify Alexander Peter/Millennia@Millennia as the approver.

`Routing type`

- Select “Serial” if you want each draft editor to approve the same copy of the document. The approvers will approve the document in the order specified in the list.
- Select “Parallel” if you want each draft editor to approve a separate copy of the document. The approvers will approve the document at the same time.

Select “Serial” for the Millennia application.

`Time limit`

- Select “Send approver a reminder after the time limit expires” if you want to notify the approver after a specified time.
- Select “Send originator a reminder after the time limit expires” if you want to notify the originator after a specified time.

Select “Send approver a reminder after the time limit expires” for the Millennia application and specify the number of days for notification as three.

`When should the originator be notified?`

- Select “After final approver” if you want to notify the originator only after the final approver.
- Select “After each approver” if you want to notify the originator after each approver.

Select “After final approver” for the Millennia application.

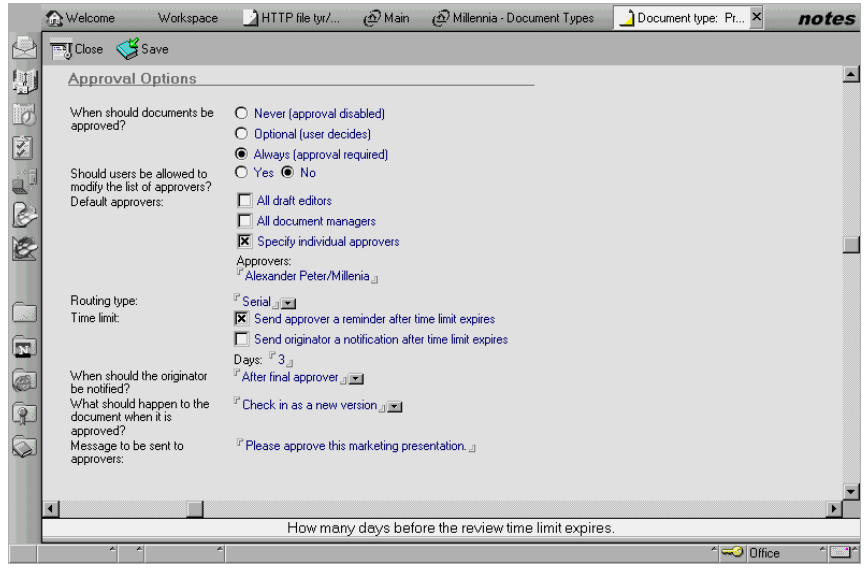
What should happen to the document after it is approved?

- Select “Check in as new version” if you want to check in the document as a new version after the final approver.
- Select “Check in and replace as current version” if you want to check in the document as the current version by replacing the existing one.
- Select “Return to the initiator” if you want to return the document to the initiator after the final approver.

Select “Check in as new version” for the Millennia application.

Message to be sent to approvers

Enter the message you want to send to approvers. This field is optional. Enter “Please approve this marketing presentation” in this field.



Archiving Options

In this section specify the parameters for automatically archiving documents that have not been previously archived.

Millennia Space Travel Agency will use Domino.Doc Storage Manager for archiving old versions of documents from the library. In this section, we discuss the archiving parameters you need to specify for archiving documents to Domino.Doc Storage Manager. Additional information about Domino.Doc Storage Manager is included in Appendix H, “Domino.Doc Storage Manager.”

The archiving parameters are:

Name of the archiving agent

Enter the name of the archiving agent in this field.

When you install Domino.Doc Storage Manager, it adds an agent called "Archive to DDSM" to the file cabinet template. Use this agent for archiving documents to Domino.Doc Storage Manager.

Domino.Doc has other example archiving agents like "Archive to File System" or "Archive to BRMS," and you can also write your own archiving agent.

If you don't have Domino.Doc Storage Manager installed you can select "Archive to File System."

Note The archiving agent name is case sensitive. Be sure to enter it exactly as it appears in the agent list of the file cabinet template.

How should documents be marked for archiving?

- Select "Do not mark documents for archiving automatically" if you want to select documents manually.
- Select "Specify an archiving trigger" if you want documents automatically marked for archiving. If you select this option, specify when documents should be marked for archiving.
- Select "Mark old versions for archiving" if you want to mark for archiving all versions of a document except the current version.

If you select "Specify an archiving trigger," specify when documents should be marked for archiving:

- Select "Archive documents older than the specified number of days" to archive versions based on document creation date. If you select this option, enter the number of days.
- Select "Archive documents that have not been modified in the specified number of days" to archive versions that have become inactive. If you select this option, enter the number of days.
- Select "Archive documents based on a formula" if you want a more sophisticated archiving process, for example, to combine some of the trigger options such as archiving older versions if they have not been modified in 30 days. If you select this option, enter the formula for archiving the documents.

Enter a valid formula that evaluates to True or False. The formula can contain any @function except the following: @Command, @DbManager, @DbName, @DbTitle, @DDEExecute, @DDEInitiate, @DDEPoke, @DDETerminate, @DialogBox, @PickList, @PostedCommand, @Prompt, and @ViewTitle.

Select "Mark old versions for archiving" for the Millennia application.

`What should happen to drafts when a version is archived?`

- Select "Leave drafts alone" if you want to leave the drafts while archiving a version.
- Select "Archive drafts when version is archived" if you want to archive drafts also while archiving a version.
- Select "Delete drafts when version is archived" if you want to delete all drafts while archiving a version.

Select "Archive drafts when version is archived" for the Millennia application.

`Allow archiving of the current version?`

- Select "Yes" if you want to allow archiving of the current version also.
- Select "No" if you do not want to allow archiving of the current version.

Select "No" for the Millennia application.

`Allow manual archiving?`

- Select "Yes" if you want to allow manual archiving.
- Select "No" if you do not want to allow manual archiving.

Select "No" for the Millennia application.

`Subform for agent-specific parameters`

Enter the name of the subform for entering archiving agent parameters, then click Specify Parameters and enter the information specific to the document type. This is an optional field.

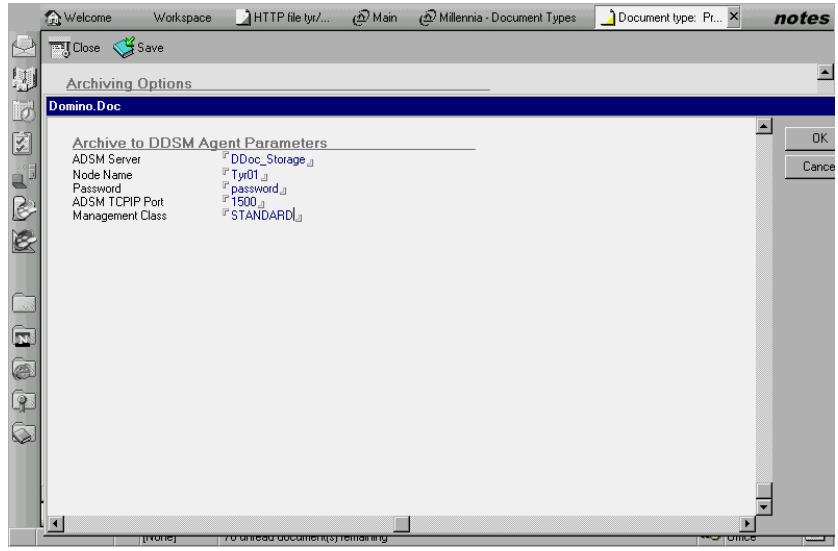
For example, if you are using Domino storage system, enter the server and storage profile for this document type; if you are using a relational database system, enter the server and instructions on how to connect.

When you install Domino.Doc Storage Manager, it adds a subform called "DDSMArchiveParameters" to the library template. Use this subform for specifying the Domino.Doc Storage Manager details like ADSM Server, Node Name, Password, ADSM TCPIP Port, and Management Class.

Enter “DDSMArchiveParameters” as the subform name and click Specify Parameters. Below are the details we used in our example:

- ADSM Server : DDoc_Server
- Node Name : Tyr01
- Password : password
- ADSM TCPIP Port : 1500
- Management Class : STANDARD

Click OK when done.



Retrieval Options

In this section, specify the parameters for retrieving archived documents back into Domino.Doc.

We used Domino.Doc Storage Manager for archiving documents from the Millennia library. In this section, we will discuss the retrieval parameters you need to specify for retrieving documents from Domino.Doc Storage Manager.

The retrieval parameters are:

`Name of retrieval agent`

Enter the name of the retrieval agent in this field. You can use one of the Domino.Doc supplied sample agents, like “Retrieve from File System” or “Retrieve from BRMS,” or you can use your own retrieval agent.

When you install Domino.Doc Storage Manager, it adds an agent called "Retrieve from DDSM" to the file cabinet template. Use this agent for retrieving documents from Domino.Doc Storage Manager.

Note The retrieval agent name is case sensitive. Be sure to enter it exactly as it appears in the agent list of the file cabinet template.

When should retrieved documents be re-archived?

- Select "Re-archive after a specified number of days" if you want to re-archive the documents after an interval of time. If you choose this option, enter the number of days for re-archiving.
- Select "Use original trigger" if you want to re-archive the documents based on the same criteria used to archive them originally.

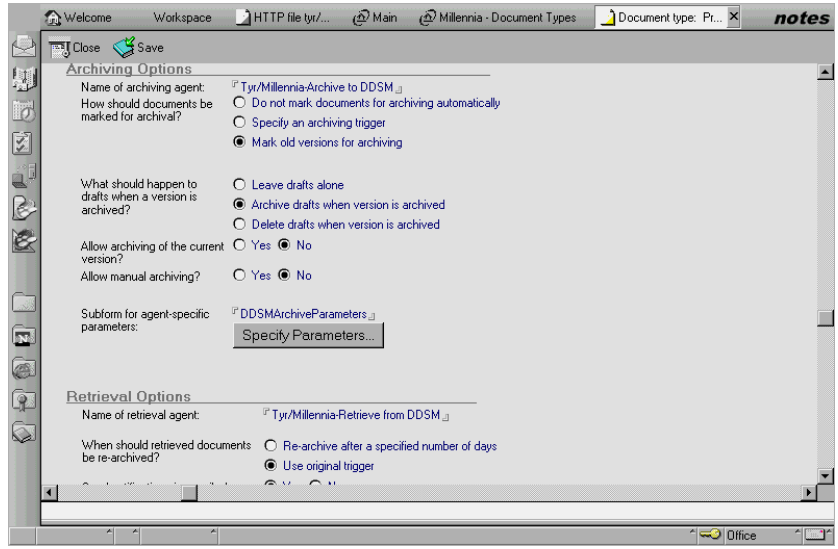
Select "Use original trigger" for the Millennia application.

Send notification via e-mail when retrieval is completed?

- Select "Yes" if you want to send an e-mail notification to the administrator when archiving is completed.
- Select "No" if you do not want to send an e-mail notification when archiving is completed.

Select "Yes" for the Millennia application.

Note Re-archiving will not automatically occur if you select "Do not mark documents for archiving automatically" under the Archiving Options.



Workflow Options

In this section, specify the parameters for using Domino Workflow with Domino.Doc. You can specify these parameters for Domino only.

Although we are not using Domino Workflow for our example Millennia application, this section describes the options you can specify when using Domino Workflow.

The Workflow parameters are:

`On which server does the workflow application reside?`

- Select "Same server as file cabinets" if the workflow application resides on the same server as the file cabinets used for workflow documents.
- Select "Other server" if the workflow application resides on another server. In this case you should specify the name of the server on which the workflow application resides.

`Specify location of workflow application`

- Select "Path" if you know the directory and file name for the Domino Workflow application database, and specify them.
- Select "By replica ID" if you know the replica ID of the Domino Workflow application database, and specify the replica ID.

`Can users initiate a new workflow process interactively?`

- Select "Yes" if you want to allow users to start workflow processes from within Domino.Doc. If you select this option, you should specify which processes are available for manual initiation.
- Select "No" if you do not want to allow users to start workflow processes.

`Which processes are available for manual initiation?`

- Select "All processes" if you want to allow users to start all workflow processes manually.
- Select "Only selected processes" if you want to allow users to start some specified processes only. If you select this option, select the processes you want the users to initiate from within Domino.Doc.

`Automatically initiate a workflow process upon the following event`

- Select "Never" if you do not want to initiate a workflow process automatically.
- Select "First document check in" if you want to initiate the workflow process when the first document is checked in to the library.

- Select “Approval cycle completion” if you want to initiate the workflow process once the approval cycle is complete.
- Select “Review completion” if you want to initiate the workflow process once the review process is complete.
- Select “Addition to specified binder(s)” if you want to initiate the workflow process when documents are added to some specified binders. If you select this option, you should specify the binders’ names.

`Which process should be initiated upon this event?`

Select the workflow process you want to initiate upon the event you specified.

`What is the default workflow process name?`

- Select “Document Title” if you want document title as the default workflow process name.
- Select “User name & Date” if you want user name and date as the default workflow process name.
- Select “Content of selected field” if you want the content of a selected field as the default workflow process name. If you select this option, specify the field name.

`What is the default priority?`

Select the default priority as High, Medium, or Low.

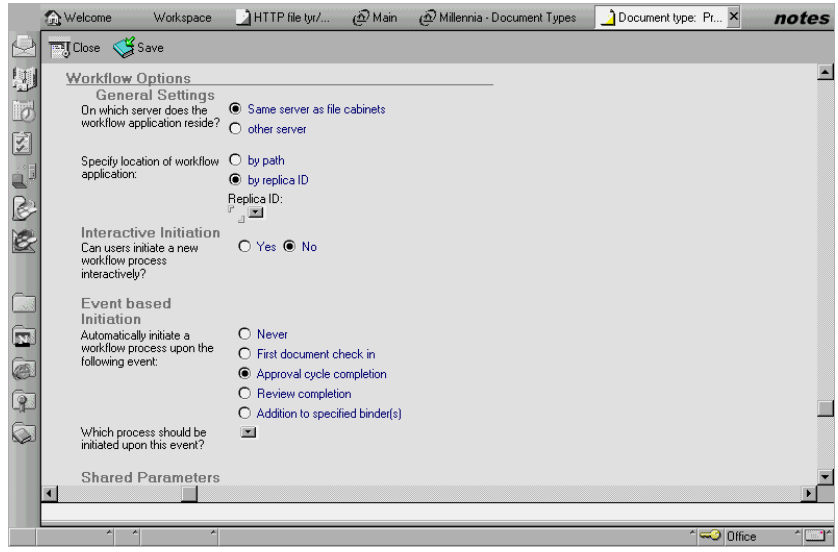
`As what type of document are the references to Domino.Doc documents added?`

- Select “Main Document” if you want the reference document to be the main document.
- Select “Binder Document” if you want the reference document to be a binder document.

`Which profile fields should be transferred to the reference document?`

- Select “No fields” if you do not want to transfer any profile fields to the reference document.
- Select “All fields” if you want to transfer all profile fields to the reference document.

- Select “Selected fields” if you want to transfer some selected profile fields to the reference document. In this case, you should select the profile fields.



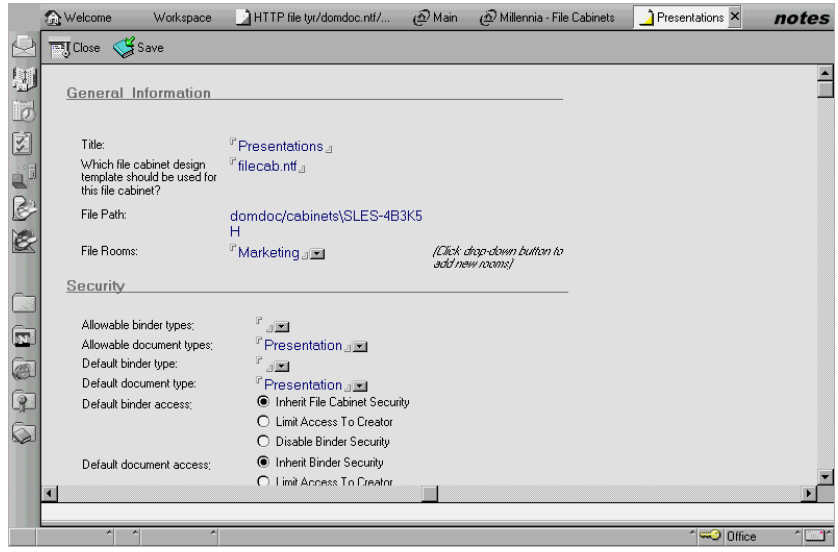
Applying the document type to a file cabinet

After creating the new document type, you have to include it in the file cabinet definition so that users can create documents of this type within the file cabinet.

To include the new document type in the file cabinet, follow these steps (we assume that a file cabinet named Presentations already has been set up in Domino.Doc):

1. Open the Domino.Doc library from a Notes client by double-clicking the database icon.
2. Click Library Administration on the main navigator.
3. Select the Presentations file cabinet from the list and double-click to open it.
4. Put the document in Edit mode by clicking Edit.
5. Include Presentation in Allowable document types.

6. Select Presentation as the default document type.



7. Save the changes and close the document.

In the previous sections, we discussed designing a subform called Presentation, creating a Presentation document type based on this subform, and then applying this document type to the Presentation file cabinet. The marketing presentations of Millennia Space Travel Corporation will be of the Presentation document type.

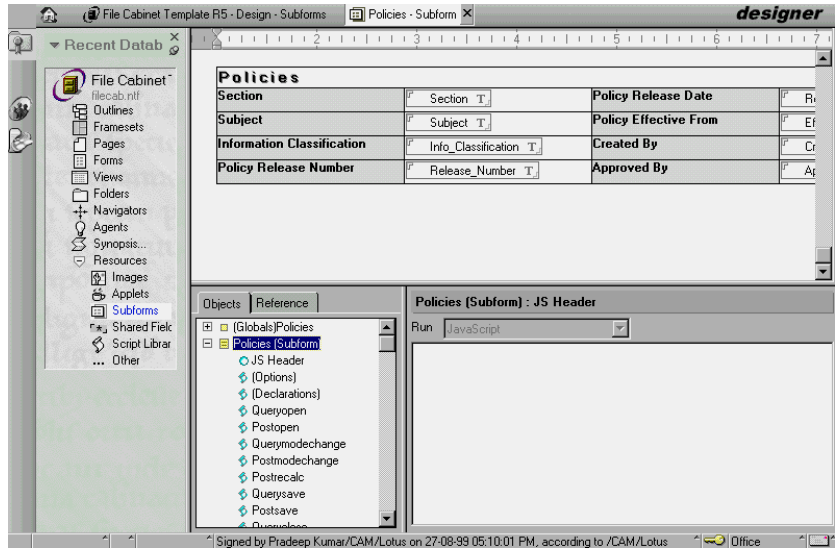
Another case study document type

In our case study we added one more document type to hold scanned documents with law information from the different planets where Millennia Space Travel Agency operates.

Follow similar procedures to create a document type called “Policies” for the Millennia application and apply it to the Interplanetary Policies file cabinet. This document type is for creating policy documents for different planets. The Policies subform should have the following fields and field definitions:

Field name	*Type*
Section	Text, Editable
Subject	Text, Editable
Info_Classification	Text, Editable
Release_Number	Text, Editable
Release_Date	Date/Time, Editable
Effect_From	Date/Time, Editable
Created_By	Names, Editable
Approved_By	Names, Editable

The Policies subform will look like this:



Defining a binder type

A binder type associates a set of profile attributes with the binders created with that type. You can make any or all of these attributes required, thus preventing the user from saving a new binder of this type without entering necessary profile information. Domino.Doc comes with a set of binder types. Additionally, you can create your own binder types.

To create a new binder type, complete the following task:

- Design the binder type subform in the file cabinet template.
- Create a new binder type in the library.
- Apply the binder type to a file cabinet.

The detailed instructions for each task are presented in this section.

Designing the binder type subform

The first step in creating a new binder type is to add a subform to the file cabinet template. The fields and formulas in the subform will be the attributes in the binder profile.

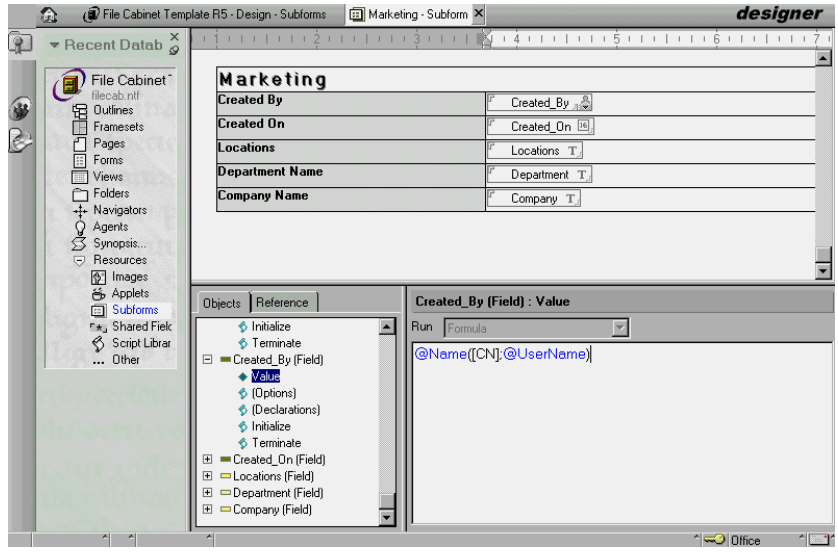
We will design a subform called "Marketing" for Millennia Space Travel Corporation.

To design a subform, perform the following steps:

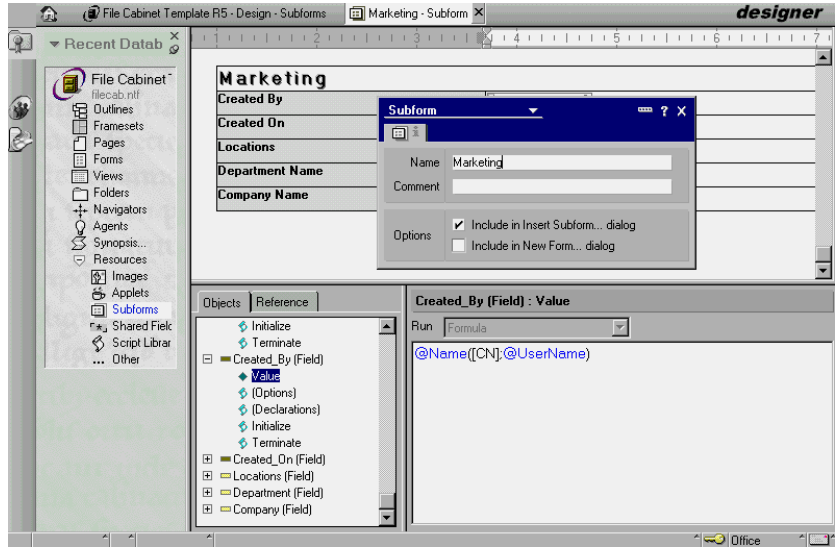
1. Open the File Cabinet Template (filecab.ntf) in Domino Designer and choose Create - Design - Subform.
2. We will now add fields to the subform. You do this by selecting Create - Field… . This places a new field on the subform at the cursor location and opens the Properties box for the field where you specify field name, type and so on.

   Add the following fields, with the field attributes given, to the subform:

Field name	*Type*	*Other options that need to be changed*
Created_By	Names, Computed when composed	Programmers pane: For the value of Created_By field, enter the formula `@Name([CN];@UserName)`
Created_On	Date/Time, Computed when composed	Programmers pane: For the value of Created_On field, enter the formula `@Today`
Locations	Text, Editable	Allow multiple values
Department	Text, Editable	
Company	Text, Editable	



3. Select Design - Subform Properties. The Properties dialog box opens.
4. In the Name field, enter Marketing. (The subform name will be the name of your new document type.)
5. Ensure that the option "Include in Insert Subform....dialog" is checked.



6. Save and close the subform.

Note Be careful if you also want to add validation formulas for your fields. If a validation formula is incorrect, the user may not be able to save a new document based on this subform.

Creating a categorized binder subform

For a binder type subform, you can use the field called BinderCategory to provide a way for users to categorize binders. In this field, subcategories can be created by entering a \ (backslash) character between the category and subcategory; for example Training\Compensation. If you want the binder creators to select from a predefined list of categories, make your BinderCategory field a keyword list. Use a multi-valued field if you want to allow a binder to appear in more than one category.

Creating a binder type in a library

After creating the binder type subform in the file cabinet template, add the new binder type to the library.

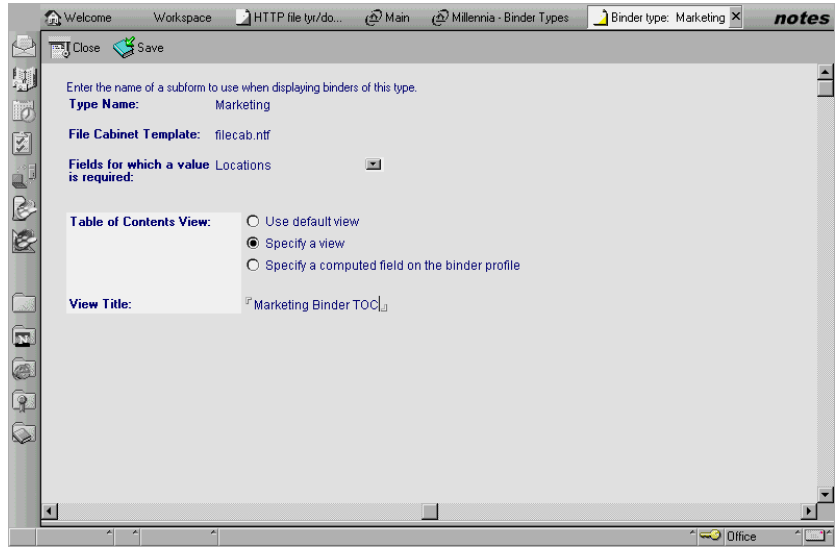
To do this, perform the following steps:

1. Make sure that the design changes you have made to the File Cabinet Template have been applied to the file cabinet databases. You can start the design update process for all databases on the Domino server by typing the following command on the server console

   ```
   load design
   ```

   Wait until the design task has completed.
2. Double-click the database icon to open the Domino.Doc library.
3. Click Library Administration on the main navigator.
4. Click Create Binder Type. The new binder type form is displayed.
5. Complete the new binder type form as follows:
   - Enter the binder type name as Marketing (the name of the binder type subform you created).
   - Enter the file cabinet template name as filecab.ntf. The default value of this field will be filecab.ntf. You can change this value to the template name you are using for your application.
   - Select Locations as a required field.
   - Select Specify a view, and enter the view name as Marketing Binder TOC.

The completely filled binder type form looks like this:



6. Save and close the new binder type form.

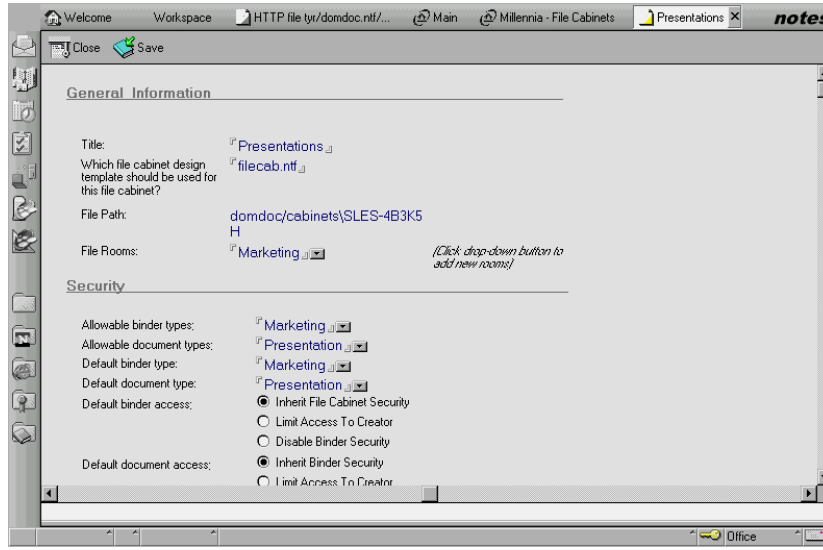
Applying the binder type to a file cabinet

After creating the new binder type, you have to include it in the file cabinet definition, so users can create binders of this type within the file cabinet.

To include the new binder type in the file cabinet, follow these steps:

1. Double-click the database icon to open the Millennia library.
2. Click Library Administration on the main navigator.
3. Select the Presentations file cabinet from the list and double-click to open it.
4. Put the document in Edit mode by clicking Edit.
5. Include Marketing in Allowable binder types.

6. Select Marketing as the default binder type.



7. Save the changes.

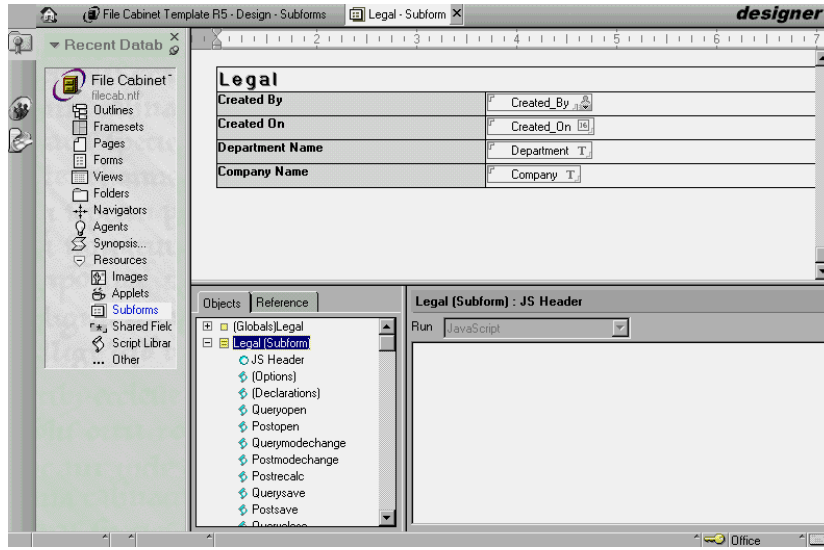
Caution Do not embed Web elements (for example, embedded navigator and file upload control) in a file cabinet template subform. Domino.Doc cannot ensure predictable results for these elements. Also, when naming the fields on the subform, do not use any of the field names defined by Domino.Doc.

In the previous sections, we discussed designing a subform called Marketing, creating a Marketing binder type based on this subform, and then applying this binder type to the Presentations file cabinet. The Marketing binder type is for creating binders for holding marketing presentations

Follow similar procedures to create a binder type called “Legal” for the Millennia application and apply it to the Interplanetary Policies file cabinet. The Legal binder type is for creating binders for holding policy documents of different planets. The Legal subform should have the following fields and field definitions:

Field name	*Type*	*Other options that need to be changed*
Created_By	Names, Computed when composed	Programmers pane: For the value of Created_By field, enter the formula `@Name([CN];@UserName)`
Created_On	Date/Time, Computed when composed	Programmers pane: For the value of Created_On field, enter the formula `@Today`
Department	Text, Editable	
Company	Text, Editable	

The Legal subform should look like this:



Rebuilding folders

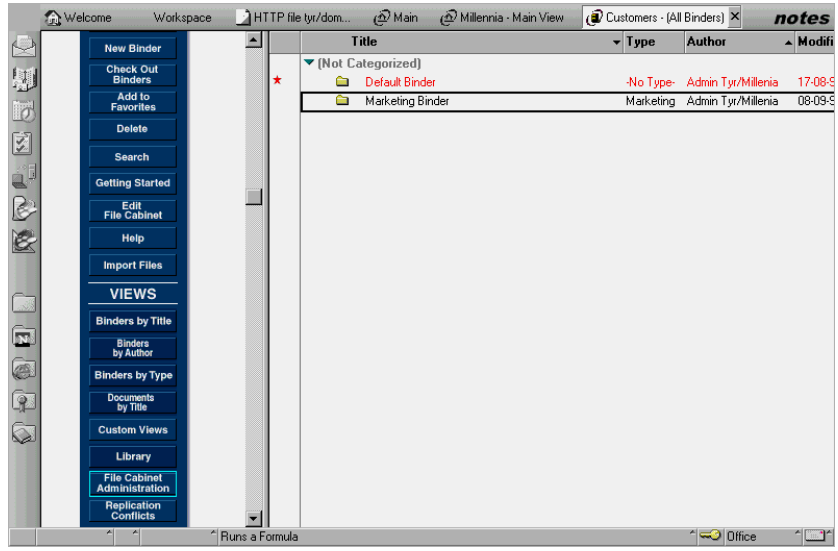
If you are using Notes folders for the binder TOC, there are some cases when you should rebuild the folders. Several of these instances are as follows:

- If you modify the design of binder TOC view, you must rebuild all folders for binders of this type.
- If you choose a different TOC view for a binder type - that is, if you open the binder type document in the library and change the TOC view field - you must rebuild all folders for binders of this type.
- If a binder contains multiple versions of the same document, the binder may have become corrupted. To correct this, you should rebuild the folder for this particular binder.

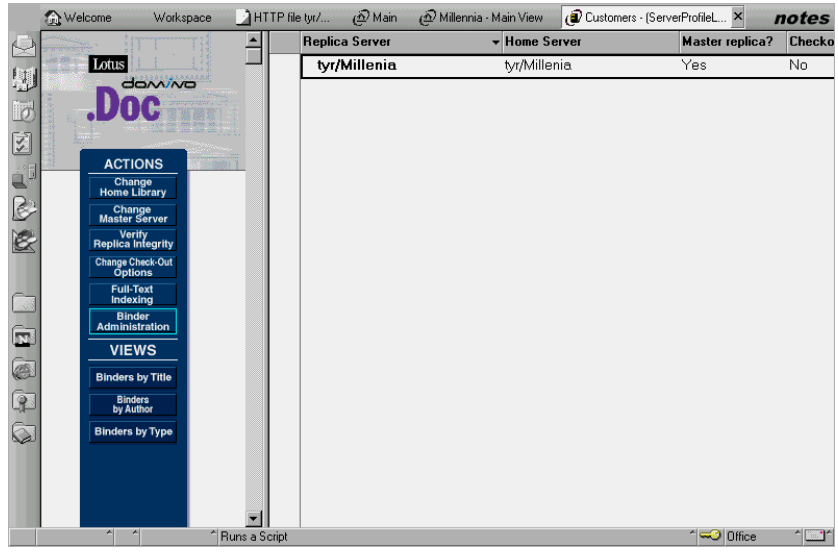
To rebuild the folders, perform the following steps:

Note This procedure must be done through Domino, not through a browser.

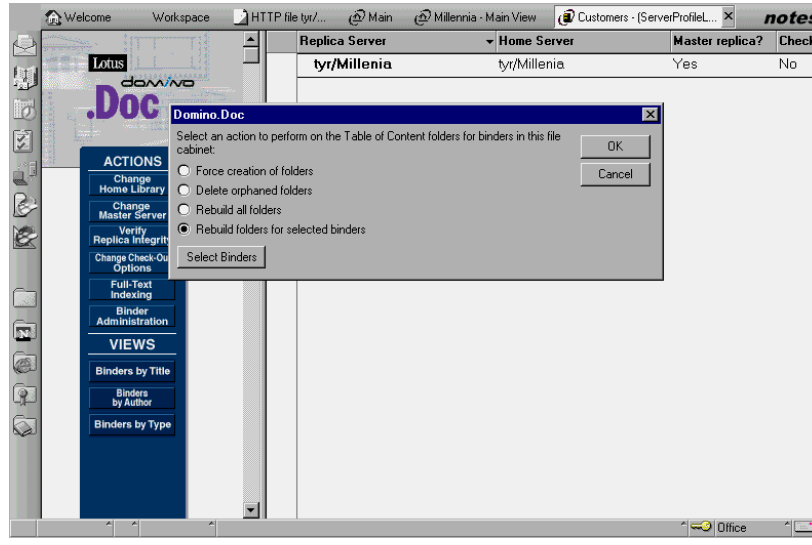
1. On the master replica server, open the library database.
2. Select the file cabinet containing the binder whose folder you want to rebuild, and double-click to open it.
3. Click File Cabinet Administration.



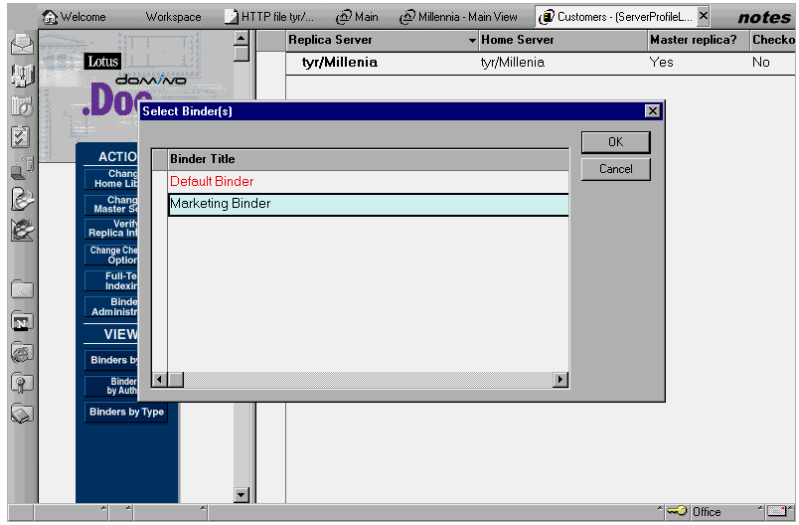
4. Click Binder Administration. A dialog box is displayed.



5. Take *one* of these two steps:
   - To rebuild all folders, select Rebuild all folders and then click OK.
   - To rebuild selected folders, select Rebuild folders for selected binders, and then:



a. Click Select binders. The Select Binder dialog box displays a list of binders.



b. Select the binders you want to rebuild and click OK. The Select Binder dialog box closes.

c. Click OK.

6. Repeat steps 1 through 5 for each file cabinet that contains binders to be rebuilt.

Note If you have difficulty rebuilding folders, the workaround is to manually delete all folders from every document database in the file cabinet, then force creation of all folders for that file cabinet using "Binder Administration."

Modifying document and binder subforms

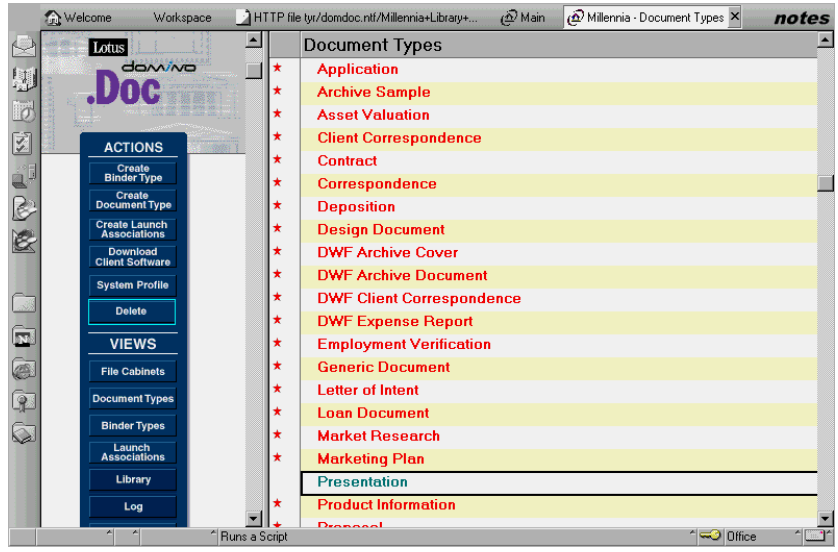
You can modify the document and binder types by editing the appropriate subforms in the file cabinet template. You can add or delete fields, change the validation formulas for a field, or make simple cosmetic changes. The changes to the subform will not be reflected in the existing documents/binders of this type or the document/binder revision history.

Deleting document and binder types

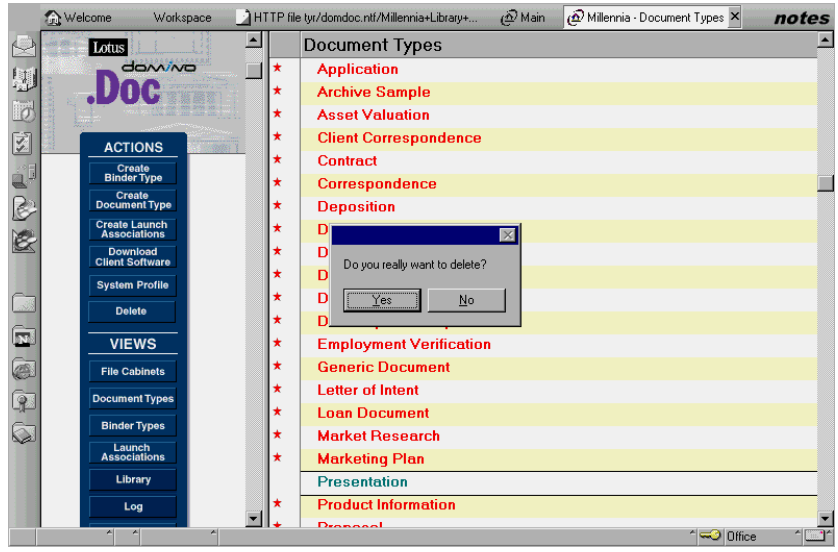
To delete a document or binder type, perform the following steps:

1. Open the library database.
2. Click Library Administration.
3. Click Document Type or Binder Type, as appropriate.

4. Select the appropriate document or binder type.
5. Click Delete. The Delete confirmation box is displayed.



6. Click Yes.



7. Click File Cabinets.
8. Open each file cabinet document and take one of following steps, as appropriate:
   - Delete the binder type from the Allowable Binder Types and Default Binder Type fields.
   - Delete the document type from the Allowable Document Types and Default Document Type fields.

Note The subform for this document or binder type will remain in the file cabinet template so that existing documents or binders of this type are not affected. Additionally, information about the type remains in the configuration database.

Note After modifying the design of a template database, you should refresh the design of databases created with that template for the changes to become effective. To refresh the database design choose File - Database - Refresh Design, or enter the Load Design command at the server console.

Summary

In this chapter, we discussed document and binder types, the steps to create a new document and binder type in the library, including designing the subforms for document and binder type and applying them to the file cabinets. We also discussed the field type considerations a designer must know when designing document type subforms for ODMA-enabled applications, and how to categorize binder subforms.

Finally, we discussed rebuilding binders, modifying document and binder subforms, and deleting document and binder types.

Chapter 5
Customizing forms

This chapter introduces the topic of customizing forms within Domino.Doc. We will discuss document-related and search-related forms in Domino.Doc and give examples of how to customize them.

What is a form

Forms are design elements in Domino databases. Users utilize forms to create and view or edit information in Domino databases. Forms contain a variety of design elements: fields to store and display information, static text, buttons, sections, subforms, and so on. You can also use forms to get input from the user without storing it in a Domino database. An example of this is whan a user performs a search by entering their search text in a field on a form and clicking a button on the form to execute the search.

Most often in Domino.Doc several databases are based on the same template. When you modify a form you should do it in the template. Once you have tested your changes you must refresh the design in all the databases that use that template. You can do this by typing the following command on the Domino server console:

```
Load design
```

Forms in Domino.Doc

There are many forms in Domino.Doc that can be modified. The two most common sets of forms, the ones pertaining to the Domino.Doc documents and the ones pertaining to searching, are discussed in detail later in this chapter.

Refer to Appendix A, “Domino.Doc forms,” for a list of all the forms found in the library and file cabinet templates.

Some typical changes to forms that you may want to perform are adding extra fields, and changing static text to conform to the naming conventions used in your company’s document management policy. Other changes can be to the color of the form, the font type and size, and so on.

Some forms of particular interest in the library template are NotesSearch and Sample Notes App Integration. The NotesSearch form is discussed later in the section "Modifying search-related forms."

The Sample Notes App Integration form is useful because it contains some examples of code used for integrating Domino.Doc with other applications. One example that is included is a "Send to Domino.Doc" button that can be added to any Domino application (including e-mail), so that any Domino document can be added to a Domino.Doc library.

Modifying document-related forms

This section discusses modifying the forms related to Domino.Doc documents, specifically the following from the file cabinet template:

- NewDocument

  This form is used when a document is created, to specify information about that document. The only time this form is shown is when a new document is created.

- Document

  This form is used to display information about existing documents.

Two different forms are used to enter and display information for the same document because the user has different data entry and functionality needs depending on whether the document is new or already exists in Domino.Doc. There may be some information that the user should not be allowed to change once the document has been saved; some information, like version number or life cycle status, does not make any sense for a new document; and so on. By using two forms, it is much easier to distinguish between what the user can do for a new document and for an existing document.

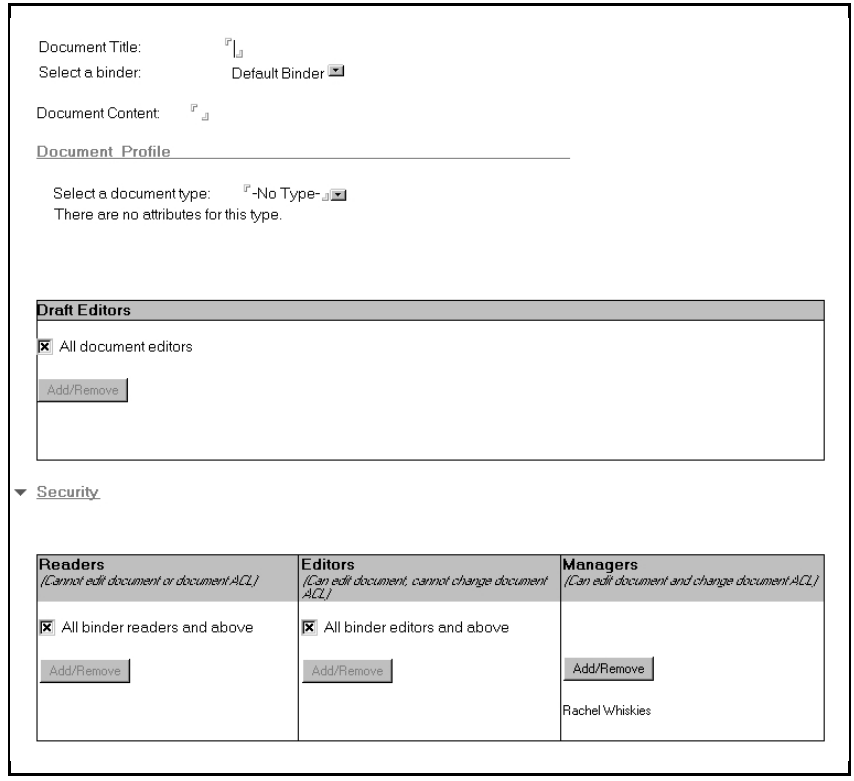
If you want to customize which information to gather and display for documents, you most often need to modify both the NewDocument and the Document form.

You can modify any other document-related forms using the same general steps described in the section below.

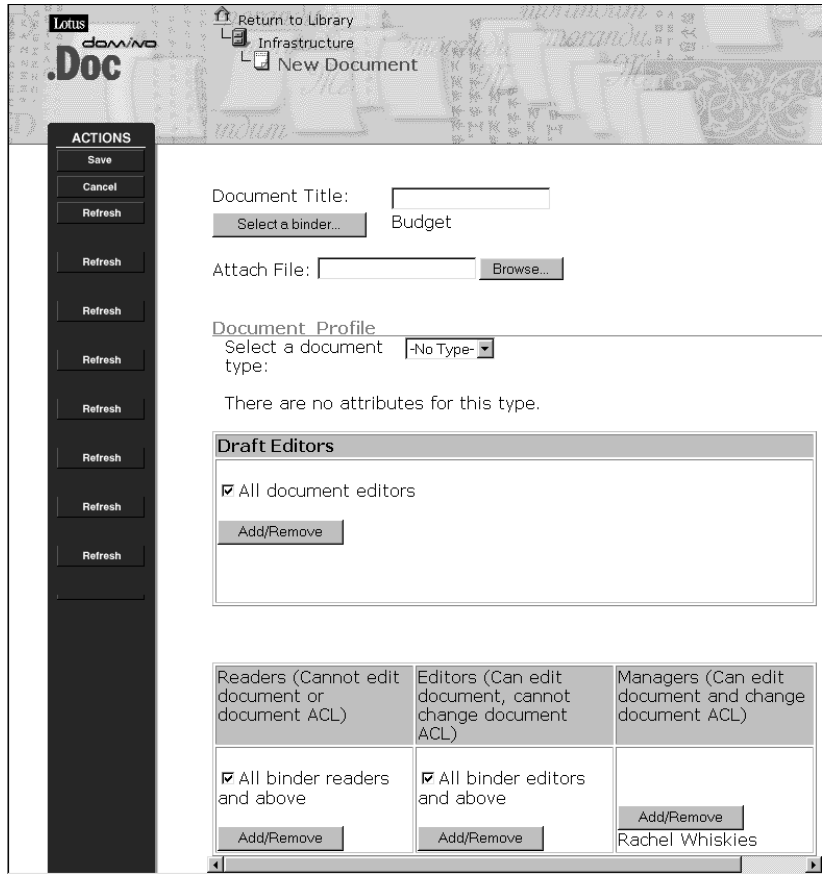
NewDocument

The NewDocument form is used when a user creates a new Domino.Doc document, whether from the Notes client or from a Web client. To change the look and feel of how information is captured about documents for the Domino.Doc library, you modify this form.

The following figure shows the NewDocument form as seen by the end user from within the Notes client:



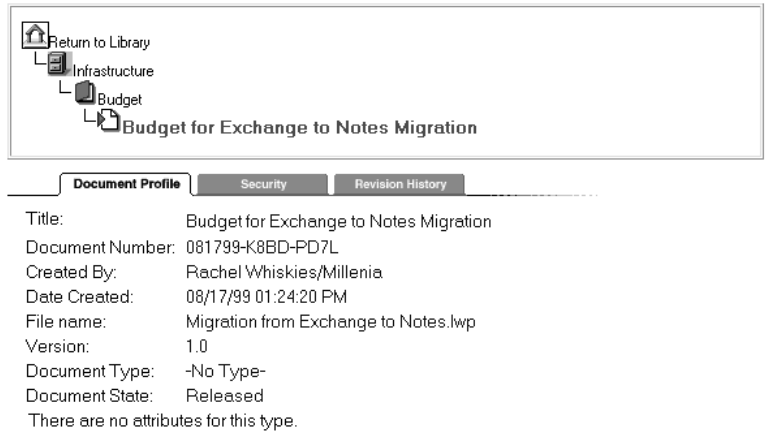
The following figure shows the NewDocument form as seen by the end user from within the Web client:



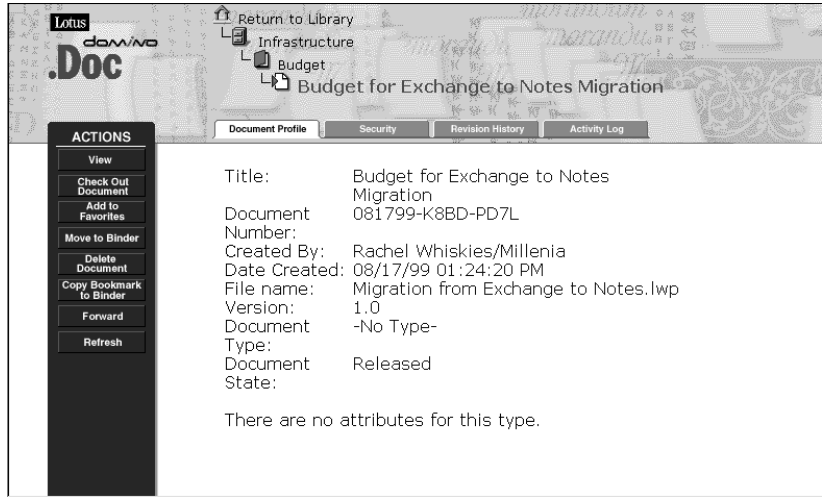
Document

The Document form is used to show Domino.Doc documents (not to be confused with an attached document within the Domino.Doc document). Essentially, the Document form is like an index card, which contains detailed information about a document, and to which the actual document is attached. When a user selects a document from within the library, they are brought to a document based on this form, which shows attributes of the attached document like title, type, version life cycle state, and so on.

The following figure is an example of what a typical Domino.Doc document (based on the Document form) looks like from a Notes client:



The following figure is an example of what the same Domino.Doc document looks like from a Web client:



Customization

In order to produce a format that meets the requirements of Millennia Space Travel Corporation, the two forms shown previously must be modified.

Cosmetic changes such as a different color scheme are needed, as well as the addition of a field that is used by the views for counting the documents.

Change of colors

The following steps demonstrate making the required cosmetic changes to the NewDocument form:

1. Open the design of your File Cabinet Template in Domino Designer R5.0.

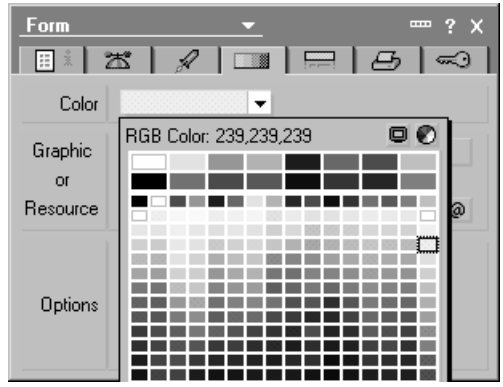
   The file name for the template is filecab.ntf. You should normally work on a server-based version of the template.

2. Select Forms in the Design pane (on the left side).
3. Open the NewDocument form in the Work pane (upper right side) by double-clicking on it.
4. Select Form Properties from the Design Menu.

   This opens the Form Properties box.

5. Select the Form Background tab, the fourth tab from the left.

   Note You can see the tab name by placing the mouse pointer over the tab; Domino will then show a small window with the tab name.



6. Modify the background color, or import a graphic to use as a watermark. (For Millennia, the background color was changed to light gray. This is shown in the above figure.)
7. Select the Defaults tab, the one with the propeller hat graphic on it.

8. Modify the colors of the Active Link, Visited Link, and Unvisited Link by selecting different colors from the drop-down box for each of the link types.
9. Save the form.

When you have tested your changes and refreshed the design of your file cabinets, your users will now see a gray background on the form they use to create new documents. If they are using a Web browser as client the colors used for active, visited, and unvisited links will also show your new selections.

If you want the same behavior for existing documents you should modify the Document form in the file cabinet template as well.

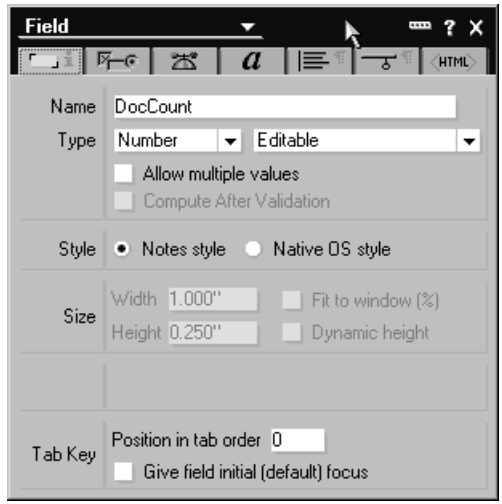
Adding a field

Add the field for counting using the following steps:

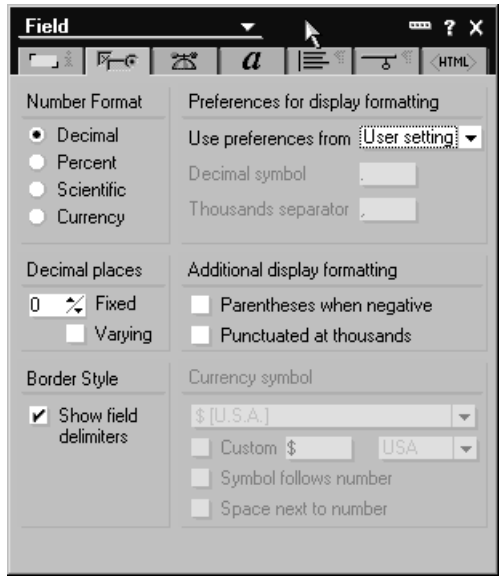
1. Open the design of your File Cabinet Template in Domino Designer R5.0.
2. Select Forms in the Design pane (on the left side).
3. Open the NewDocument form in the Work pane (upper right side) by double-clicking on it.
4. With the cursor at the top of the form, select the Create menu, then select Field . . . The field is placed at your cursor's location on the form and the Properties box for the field is displayed

   Since this field is to be hidden, it does not matter where it physically is placed on the form. However, when placing objects on a form, you should consider that calculations are done in order, starting from the top left of the page. You should also bear in mind that other developers may access your form in the future if it ever needs to be modified, which will be easier if you have a standard way of doing things. One good practice is to put all hidden fields at the very top of the page, where other developers will be able to find them easily.

5. Set the Name of the field to DocCount, and the Type to Number and Editable, as shown in the following figure:



6. Select the Control tab, the second from tab from the left.
7. Select Decimal for the Number Format, and set the number of Decimal places to Fixed and zero, as in the following figure:

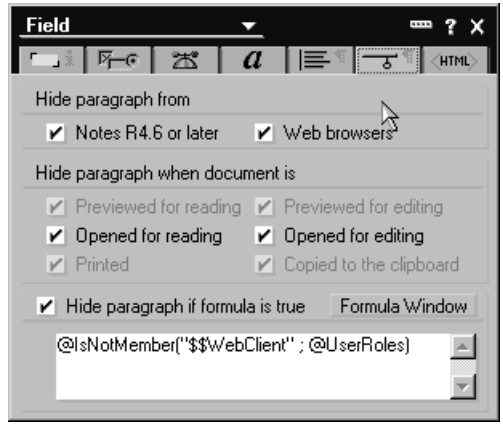


If you are using Designer for Domino R4.6 you should select Number Format to be Fixed, with zero number of decimal places.

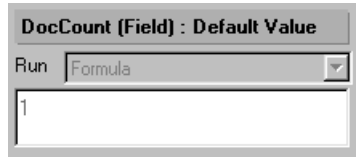
8. Select the Paragraph Hide When tab, the sixth tab from the left.

9. Set the field to be hidden at all times by selecting the following check boxes:
   - Hide paragraph from Notes R4.6 or later
   - Hide paragraph from Web browser

   as in the following figure:



10. Close the Properties box for the field by clicking the x in the upper right corner.
11. Click on the input area in the Programmers' pane in the lower right corner of Domino Designer, and give the field a default value of 1, as in the following figure:



12. Save the form.

This field was added to the form so that every Domino.Doc document will have this field. The field is hidden because, with a constant value of 1, it is not of much interest when using the form. However, because every document will have this field, the total number of documents in a view can be easily calculated by finding the sum of this field for all documents shown.

Note Documents that were created with the form before the hidden field was added will not have the field with a value of 1 and thus not be included in the count. To remedy this you can create an agent which adds a field with a value of 1 to all existing documents in the database. Read more in Chapter 9, "Agents," about how to work with agents.

While you normally would repeat the steps for the Document form you do not need to do it in this instance as the hidden field would be of no use on the Document form.

Modifying search-related forms

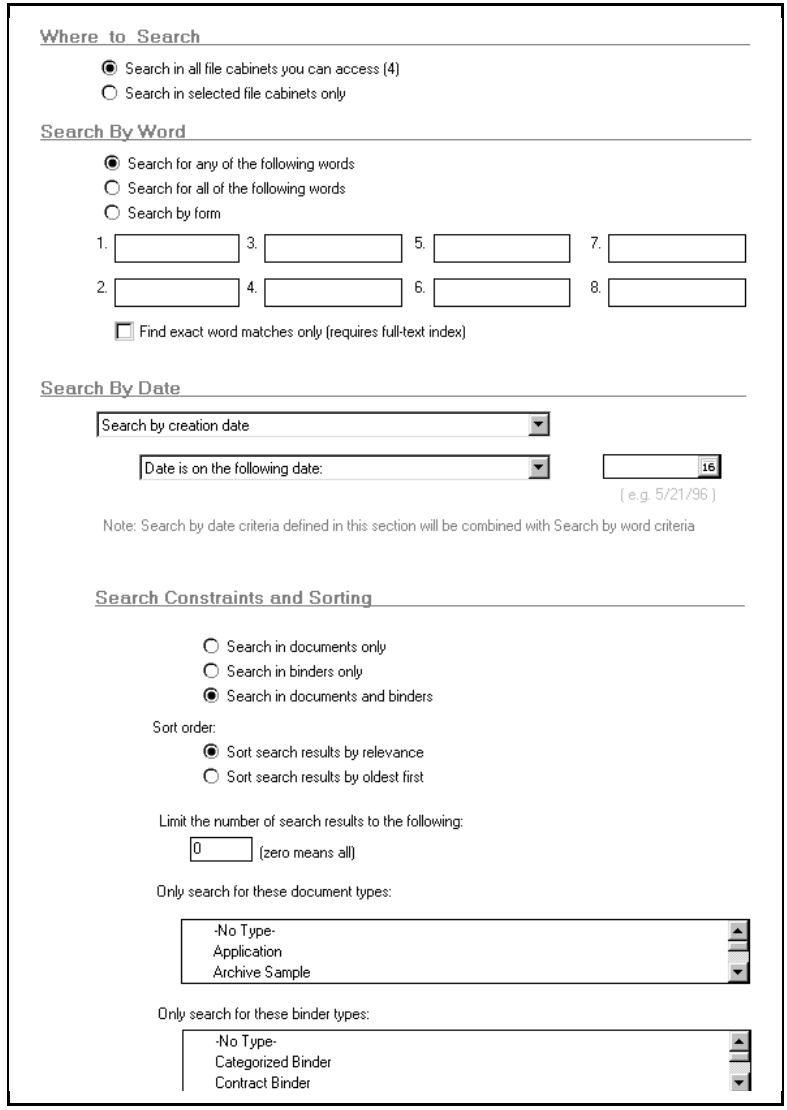
In order to customize the way users search for documents in Domino.Doc, the search-related forms must be modified. These forms include the following:

- From the library template:
  - NotesSearch: the search page seen by Notes users when they do a search from within the library database
  - WebSimpleSearchAll: the search page seen by Web users when they do a simple search from within the library database
  - WebSimpleSearch: the search page seen by Web users when they do a simple search from within a file cabinet
  - WebAdvancedSearchAll: the search page seen by Web users when they do an advanced search from within the library database
  - WebAdvancedSearch: the search page seen by Web users when they do an advanced search from within a file cabinet database
- From the file cabinet template:
  - NotesSearch: the search page seen by Notes users when they do a search from within a file cabinet database

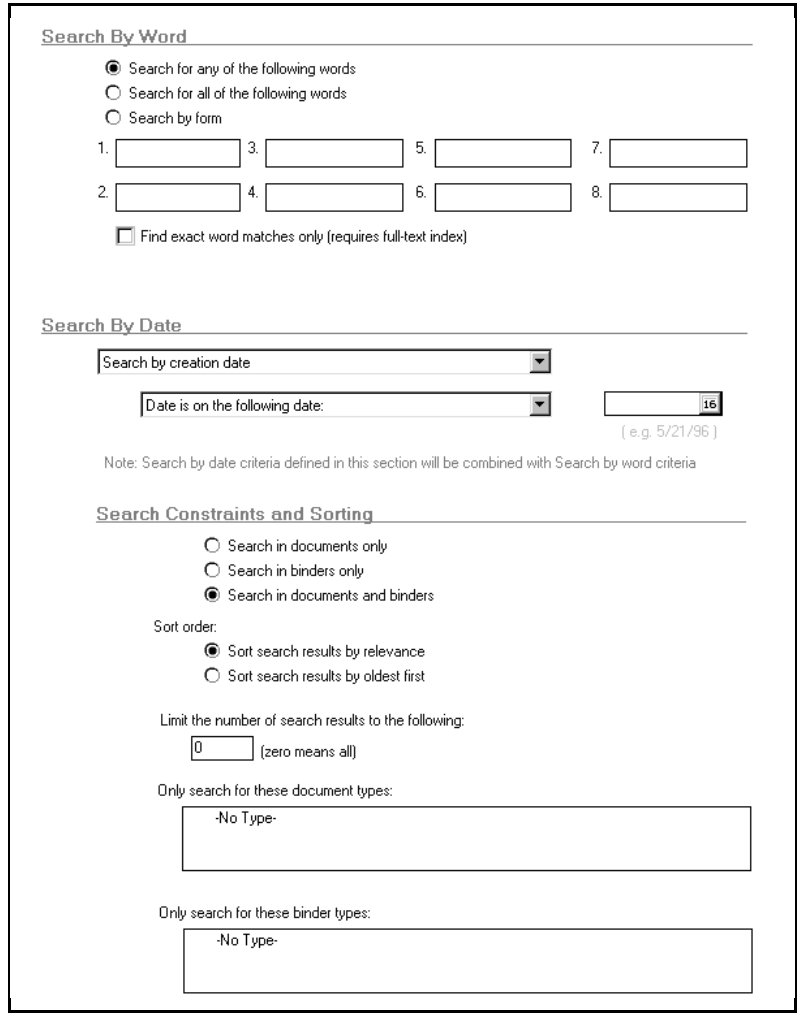
  Note Although this has the same name as the form in the library template, it is different. The form in the library template allows users to choose which file cabinets to search, whereas the form in the file cabinet template does not (since the user has already entered a file cabinet).

Notes client search forms

The following figure shows the library template's NotesSearch form as seen by the end user from within the Notes client:



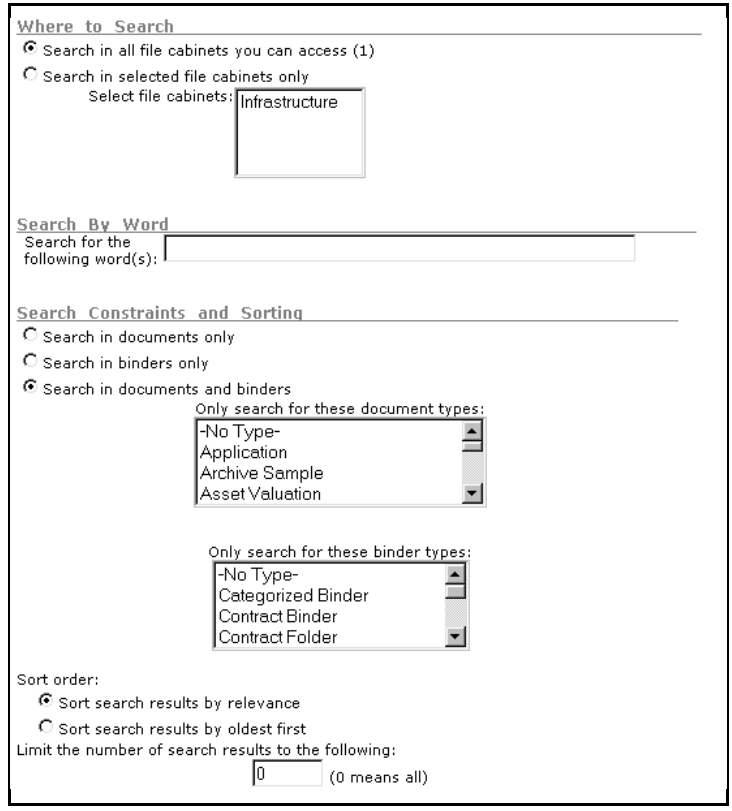
The following figure shows the file cabinet template's NotesSearch form as seen by the end user from within the Notes client:



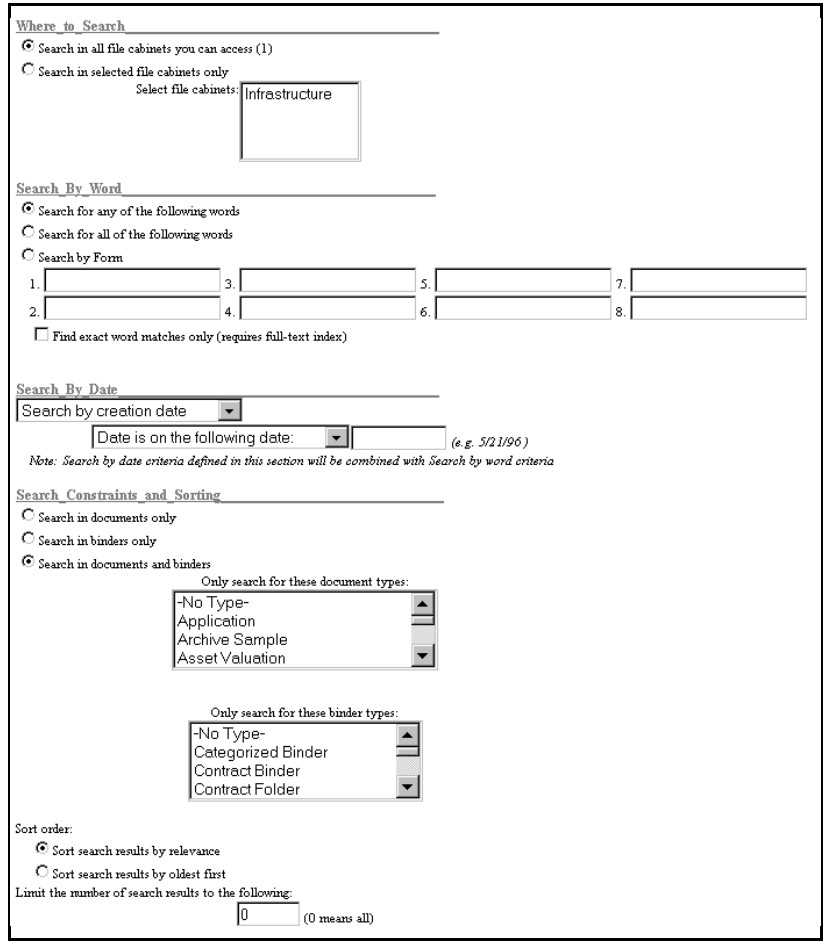
Note The end user never sees either of the above forms from within the Web client. For users of a Web client, the WebSimpleSearch, WebSimpleSearchAll, WebAdvancedSearch, and WebAdvancedSearchAll are used instead.

Web client search forms

The following figure shows the library template's WebSimpleSearchAll as seen by the end user using a Web client (note that WebSimpleSearch is the same, except it doesn't give the user a choice as to which file cabinet(s) to search):



The following figure shows the library template's WebAdvancedSearchAll as seen by the end user using a Web client (note that WebAdvancedSearch is the same, except it doesn't give the user a choice as to which file cabinet(s) to search):



Customization

This section outlines how the search-related forms are customized in order to meet the requirements of Millennia Space Travel Corporation.

The search customization required for Millennia is to add the functionality of searching for synonyms of the search request. In other words, when a user searches for a word, the search will first look in a thesaurus for other words that mean the same, and then perform a search in the library for any of the words.

This is done by adding an action to the NotesSearch forms with which users will perform a synonym search. We will first go through the actual code that looks up synonyms, and then we will go through the steps needed to put an action button with that code on a search form.

Code for looking up search synonyms

The LotusScript code looks like this:

```
Sub Click(Source As Button)
'Declare the object variables
   Dim ns As NotesSession
   Dim ndb As NotesDatabase
   Dim nw As NotesUIWorkspace
   Dim nuidoc As NotesUIDocument
   Dim nva As NotesView
   Dim ndoc As NotesDocument
   Dim ndoca As NotesDocument
   Dim CollNdoc  As notesdocumentcollection

'Declare other variables
Dim strCondition As String
Dim vntSynonym As Variant
Dim Srch As Variant
```

```
'Create the new Notes Session object
Set ns = New NotesSession

'Set the Notes Database as the current database
Set ndb = ns.CurrentDatabase

'Create a new Notes UI Workspace object
Set nw = New NotesUIWorkspace

'set the Notes UI Document to the current document
Set nuidoc = nw.CurrentDocument

'Update the Full Text Index of the current database
ndb.UpdateFTIndex False

'open the Thesaurus database
Set dbThes = ns.GetDatabase(ndb.server,"Thesaurus.nsf")

'Get the Notes Document from the Notes UI Document
Set ndoca = nuidoc.Document

'Set the variable Srch to be the contents of the
'SimpleQuery field
Srch = ndoca.SimpleQuery

'Get the Notes View from the Thesaurus database
Set nva = dbThes.GetView("VSynGeneral")

'Check to see if user entered a search criteria
If Ubound(Srch) = 0 And  Srch(0)=""  Then
      Msgbox "There is nothing to search!",64
                  ,"Thesaurus Search"
      Continue = False
      Exit Sub
End If
```

```
'Build the strCondition by adding each synonym found
Forall a In Srch
      Set CollNdoc = nva.GetAllDocumentsByKey(Srch,True)
      Set ndoc  = CollNdoc.GetFirstDocument()
      If  ndoc Is Nothing Then
            If strCondition <> "" Then
                  strCondition  = strCondition  & " OR "
            End If
            strCondition  = strCondition  & a & ""
      Else
            Do While (Not ndoc Is Nothing )
                  'Get the values from the field
                  'called txtSyn
                  vntSynonym =
                          ndoc.GetItemValue("txtSyn")
                  Forall synonym In vntSynonym
                   If strCondition <> "" Then
                    strCondition  = strCondition  & " OR "
                   End If
                  'Add the next synonym to strCondition
               strCondition = strCondition & synonym & ""
                  End Forall
                  Set ndoc =
                           CollNdoc.GetNextDocument(ndoc)
            Loop
      End If
End Forall

'Set the SimpleQuery field to the list of synonyms
Call nuidoc.fieldsettext("SimpleQuery",strCondition)
'refresh the search screen so that the new value of the
'SimpleQuery field is shown
nuidoc.refresh

End Sub
```

This code is explained below:

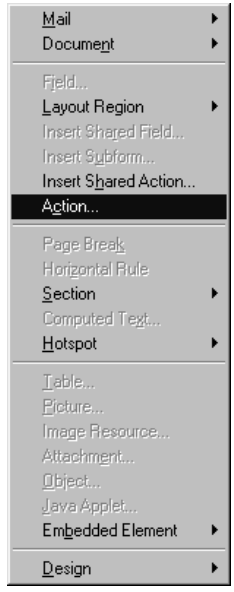
- The first section (the "Dim" statements) defines the variables that will be used in the code.
- The second section sets the actual values of the object variables, setting the variable *nuidoc* to the current open document (the search form), *dbThes* to the name of the thesaurus database (*Thesaurus.nsf*), *nva* to the name of the view in the thesaurus to check (*VSynGeneral*), and *Srch* to the value that the user entered as the search criteria.
- The third section checks to see if there actually is a value for *Srch* (if the user did not use Simple Search, or if the user did not enter a word to search, then *Srch* will be blank).
- The fourth section is the actual thesaurus lookup — it builds the value *strCondition* with the word that was first entered by the user, and then concatenates the word "OR", followed by the first thesaurus result, repeated until all of the thesaurus results are part of *strCondition*.
- The final section replaces the value of the *SimpleQuery* field with the results of the thesaurus lookup (*strCondition*).

In order to use this, there must be a database called "Thesaurus.nsf" which must contain a view called "VSynGeneral", which must contain one column made up of a field called "txtSyn". This field is simply an array consisting of a list of words that have the same meaning (for example: traveler, wayfarer, voyager, itinerant, passenger, commuter, tourist, excursionist, explorer, adventurer).

Adding an action button for synonym search

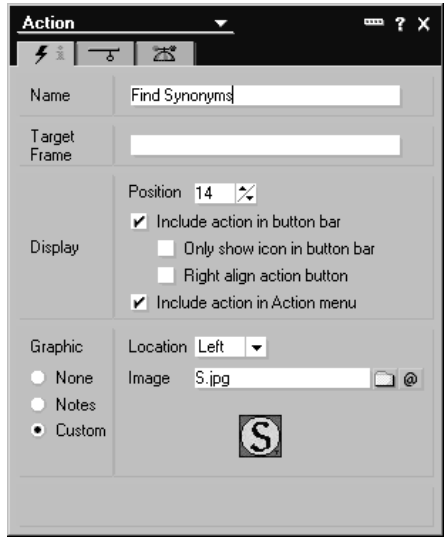
Use the following steps to add this functionality to the NotesSearch form:

1. Open the design of the Library Template (domdoc.ntf) in Domino Designer R5.0
2. Open the NotesSearch form.
3. From the Create menu, choose Action:



4. A Properties box will appear, which should be filled in with:
   - Name on the action button (is called Title in Designer for Domino R4.6)
   - Position (the default - last position is OK)
   - Include action in button bar (default)

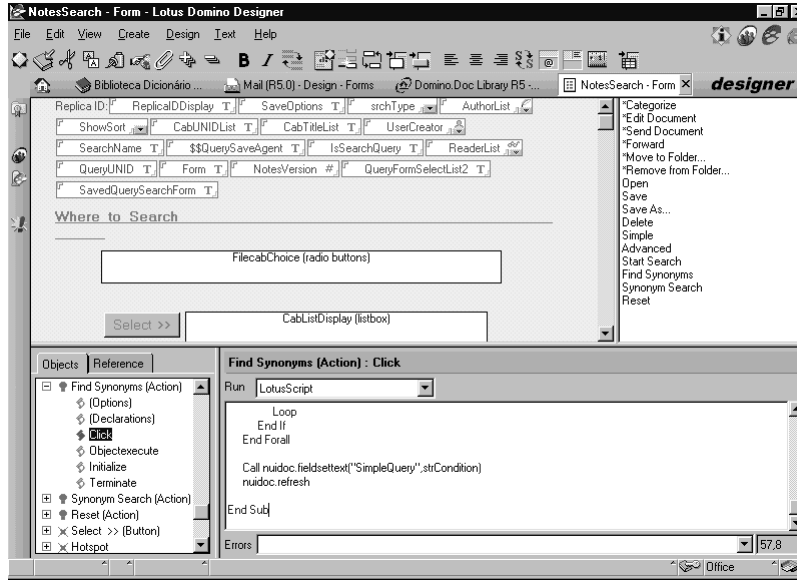
   similar to the following:



In the above figure we have also added a custom graphic for the action button. It is a file named S.jpg which shows a big S for synonyms.

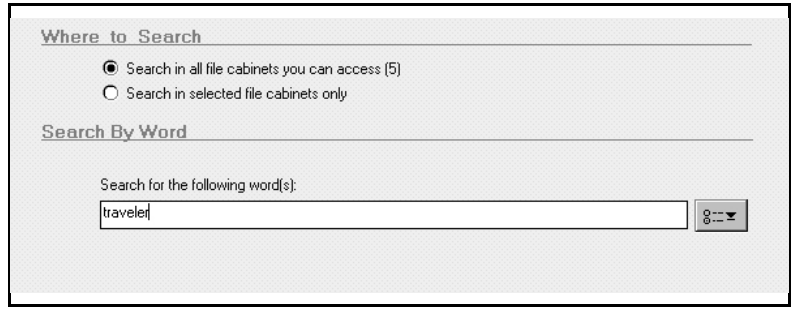
Note The capability of adding custom graphics for action buttons is not available with Designer for Domino R4.6.

5. In the programmer's pane, insert the code shown previously.

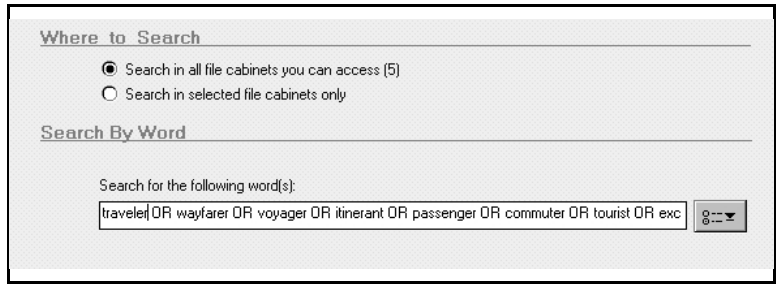


6. Repeat for the NotesSearch form in the file cabinet template.

The next figure shows a search screen that has been filled in by a user to search for references to 'traveler'.



This action now will take a word that is entered into the simple search query shown in the previous figure, and replace it with a list of synonyms from the thesaurus database, shown in the figure below.



The user then clicks the Start Search button in order for the search to actually take place.

Instead of having the user first click the button that fetches the synonyms and then another button to start the search, you can combine the two actions into one by copying all of the code from within the Start Search action subroutine (not including the Sub Click line or the End Sub line), and simply inserting it after the code for the action.

Now the user can enter one word into the simple search, click the button associated with this action, and the entire search process will take place automatically, searching for the user's word as well as for the words found in the thesaurus.

This customization can be easily modified to use the advanced search features of Domino.Doc. This is left as an exercise for the reader.

Summary

In this chapter you learned about the various forms in Domino.Doc. Some cosmetic changes were made to two forms, and a field was added to a form. Customizations to the search forms were done, as well as the addition of an action to the NotesSearch forms that allows the users to search the library based on thesaurus results.

Chapter 6
Customizing views

In this chapter, we will discuss the default Domino.Doc views, how to modify existing views, and how to create new views.

What is a view

A view is the entry point for opening and reading documents in a Domino database. It lists the documents stored in a database based on a selection criteria. Each row in a view represents data from a single document and each column represents a field or a combination of fields from that document. Thus a view can be thought of as a "table of contents" for a database.

All Domino databases have at least one view, although most databases have more than one view that organize and present documents in different ways. Different views for the same database can range from the very simple to the extremely detailed and complex, depending on the needs of the user.

Domino.Doc views

Domino.Doc comes with a set of views, for both Domino and Web users. Some of these views are used to display documents, binders, and file cabinets to users, while some of them are used by Domino.Doc for internal processing. You can see a list of the default views available in library and file cabinet templates in Appendix B: Domino.Doc views.

Modifying existing views

You can modify the default Domino.Doc views to customize your installation. Be sure to modify the views for both Domino and Web users in order to maintain a common appearance.

Important Do not modify any of the Domino.Doc internal views. This may affect the functionality of Domino.Doc. Instead, if you want, make a copy of the view and modify it.

In the previous section, we described some of the modifications you can make to the default Domino.Doc forms. In this section we will discuss modifying some of the default views to customize the software for Millennia Space Travel Corporation.

The binder TOC view is the one which you are most often required to modify as this is the view used for displaying binder contents (list of documents) unless you have chosen not to use folders. We will modify the binder TOC view and add a column to display a "NEW" tag to Web users if the document was created within the last 15 days. We will continue the example from Chapter 5, "Customizing forms," where we added a counter field to new document, and add a view column to display the total number of documents in the library. Additionally, we will make some cosmetic changes, modifying the default views to have background color and alternate row color which will give them different look.

Adding a tag to new documents for Web users

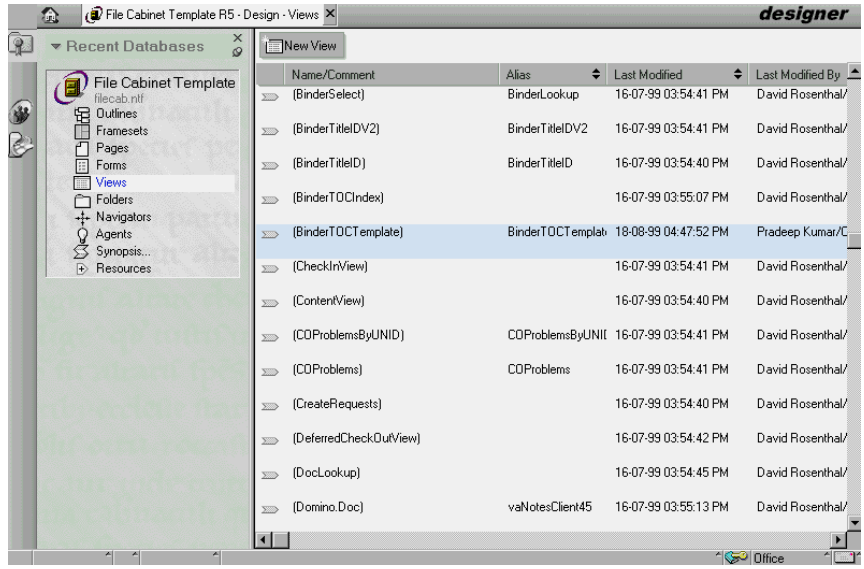
To display a NEW tag, modify the binder TOC view as follows:

1. Create a NEW tag called new.gif with one of the many third party graphics programs that can produce gif files, or get the new.gif file we used from the IBM Redbooks Website. See the appendix, "Additional Web material," at the back of this book for instructions on how to get the file.
2. Place new.gif in the domino\icons directory under the Domino data directory. The actual directory path depends on how Domino is installed. If Domino R5.0 has been installed using the default setting the directory path is

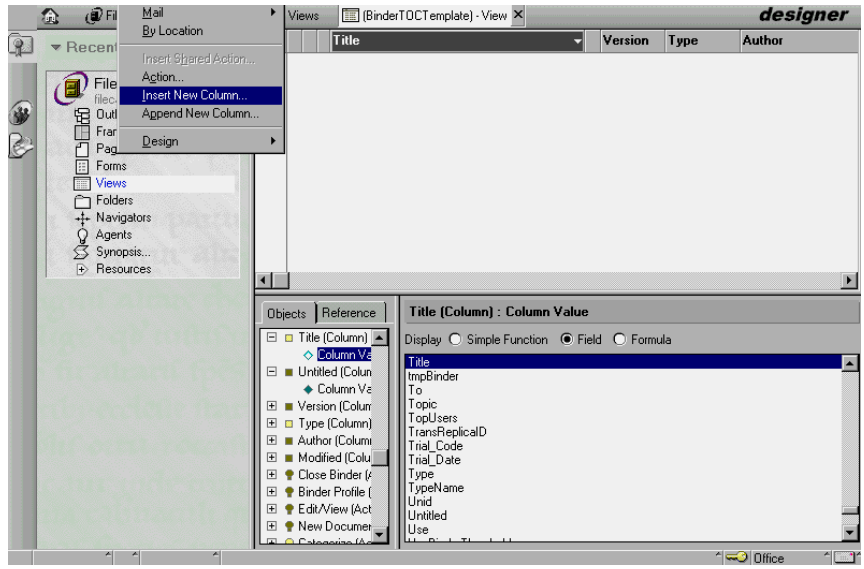
   ```
   C:\Lotus\Domino\Data\Domino\Icons
   ```

3. Open the File Cabinet Template (filecab.ntf) in Domino Designer.

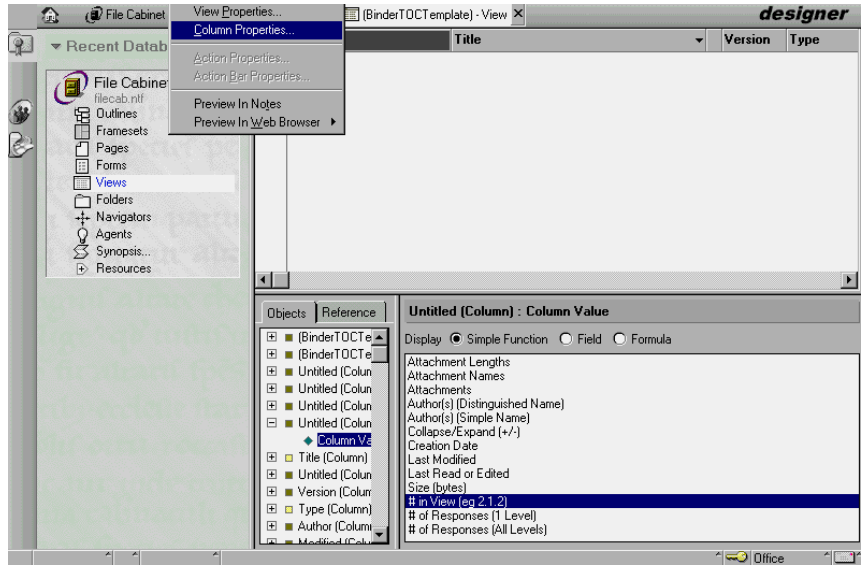
4. Select Views from the design elements list. The list of views in the template is displayed, as shown in the following figure.



5. From the list, select (BinderTOCTemplate) and double-click to open it.
6. Select the Title column and choose Create - Insert New Column.

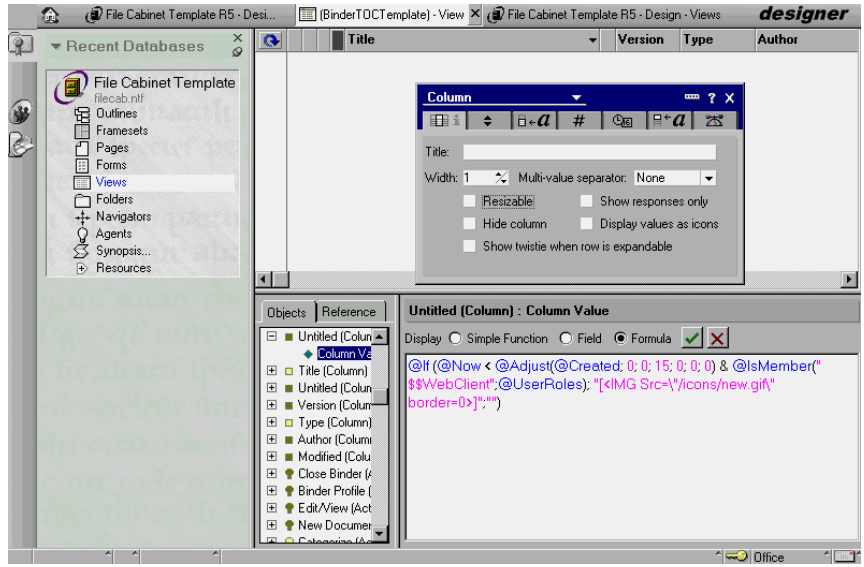


7. Select the new column and choose Design - Column properties. The Column Property box is displayed.



8. Deselect the option Resizable under Column info tab (the first tab) and keep the column width at minimum.
9. Leave the Title field blank.
10. For the column value, go to the Programmers' pane in Domino Designer and select the Formula radio button.
11. Type in the formula in the Programmers' pane entry area:

```
@If (@Now < @Adjust(@Created; 0; 0; 15; 0; 0; 0) &
@IsMember("$$WebClient";@UserRoles); "  [<IMG
Src=\"/icons/new.gif\" border=0>]";"")
```



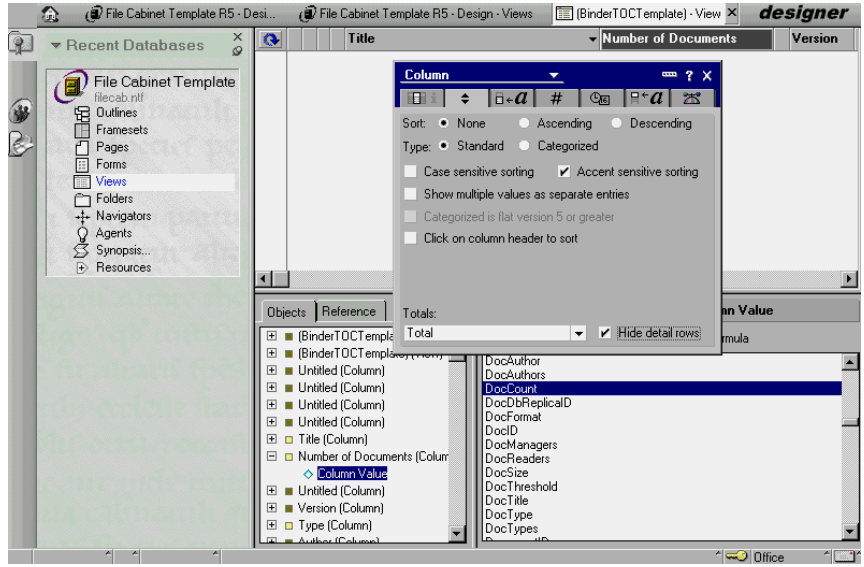
12. Save and close the view.

Adding a column to count documents

To display the total number of documents, modify the binder TOC view as follows:

1. Open the File Cabinet Template in Domino Designer.
2. Select Views from the design elements list. The list of views in the template is displayed.
3. From the list, select (BinderTOCTemplate) and double-click to open it.
4. Insert a new column after the Title column.
5. Select the new column and choose Design - Column properties. The Column Property box is displayed.
6. Type Number of Documents as the column title.
7. Under the Sorting tab, the second tab from the left, select Total in the list box under the Totals: label and select “Hide detail rows.”

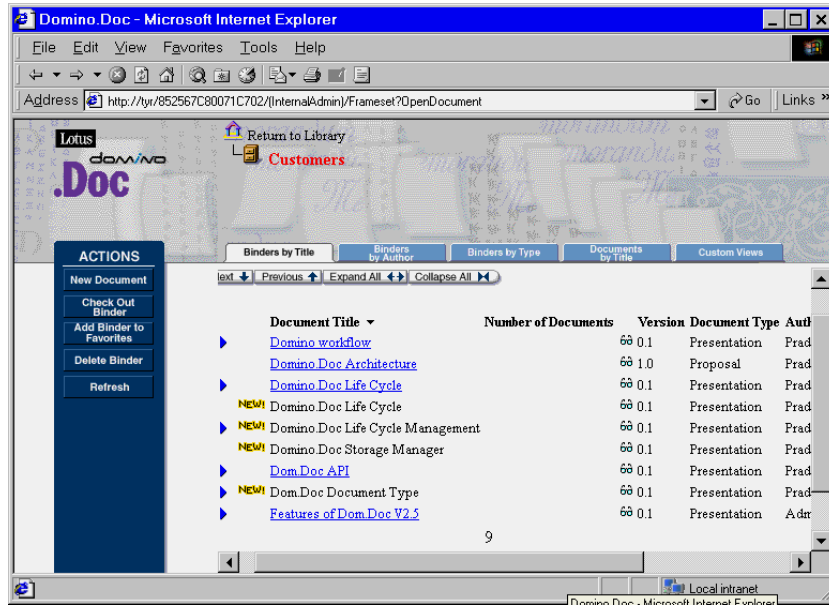
8. For the column value, select the the field radio button in the Programmers' pane and select the field DocCount (DocCount is a number field created in New Document form for document counting).



9. Save and close the view.

To put the modified view in production you must refresh the design of all your binder and document databases that inherit from filecab.ntf. After that you must rebuild your folders from within each file cabinet (File Cabinet Administration - Binder Administration - Rebuild folders).

The modified view will now look like this for a Web browser user:



Note To see this view you must have chosen to use folders instead of the BinderTOC control during the installation of Domino.Doc.

Creating a new view

In addition to the default views that come with Domino.Doc, you can create your own views in the library and file cabinet templates to suit your requirements.

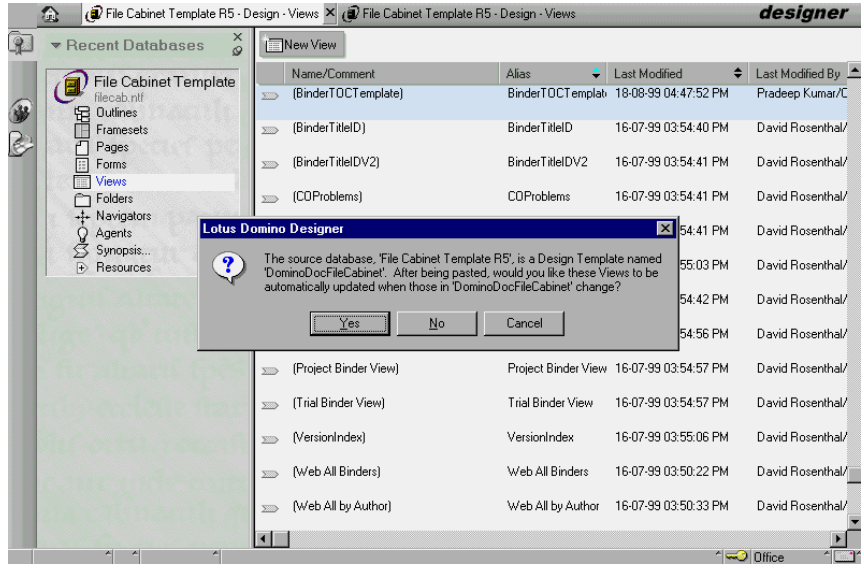
In this section, we will discuss creating a binder TOC view in the file cabinet template to customize the application for Millennia Space Travel Corporation. Also, we will create a categorized view based on travel locations for searching purposes.

Domino.Doc comes with a default view called (BinderTOCTemplate) for binder tables of contents. This template contains all of the script and action buttons that must be included in the binder TOC view.

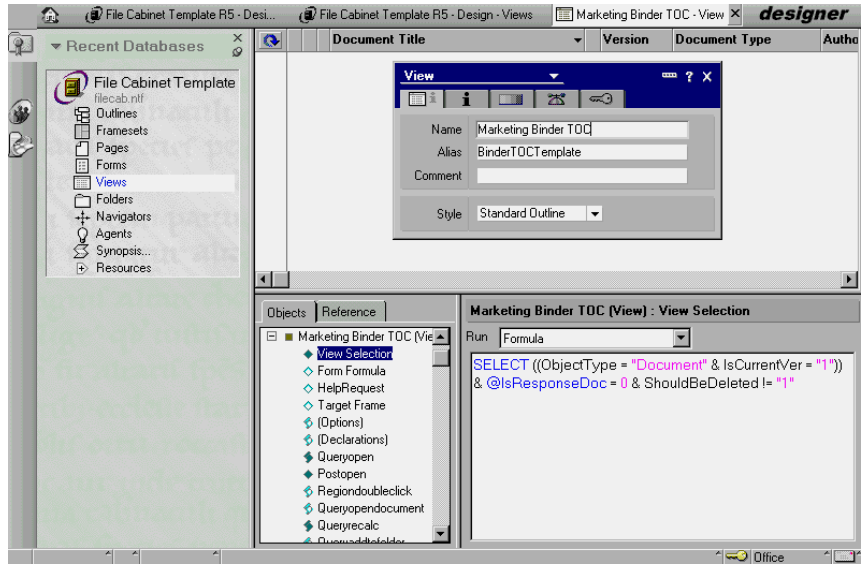
To create a customized binder TOC view for Millennia Space Travel Corporation, perform the following steps:

1. Open the File Cabinet Template in Domino Designer.
2. Select Views from the design elements list. The list of views in the template is displayed.

3. From the list, select (BinderTOCTemplate).
4. Choose Edit - Copy, then Edit - Paste. A dialog box is displayed.



5. Click Yes. The dialog box closes and a new view called Copy of (BinderTOCTemplate) appears in the view list.
6. Select this new view and double-click to open it. Choose Design - View Properties.
7. Rename this view to Marketing Binder TOC.
8. Click the Info Option tab (tab number two with the letter i) and remove the check mark beside “Default design for new folders and views.”
9. Click the Style tab (tab number three) and select a background color and alternate row color for the view.
10. Change two column headings. You do this by clicking the column heading and then changing the name in the Properties box for that column. Change Title to Document Title and Type to Document Type.



11. Save and close the view.

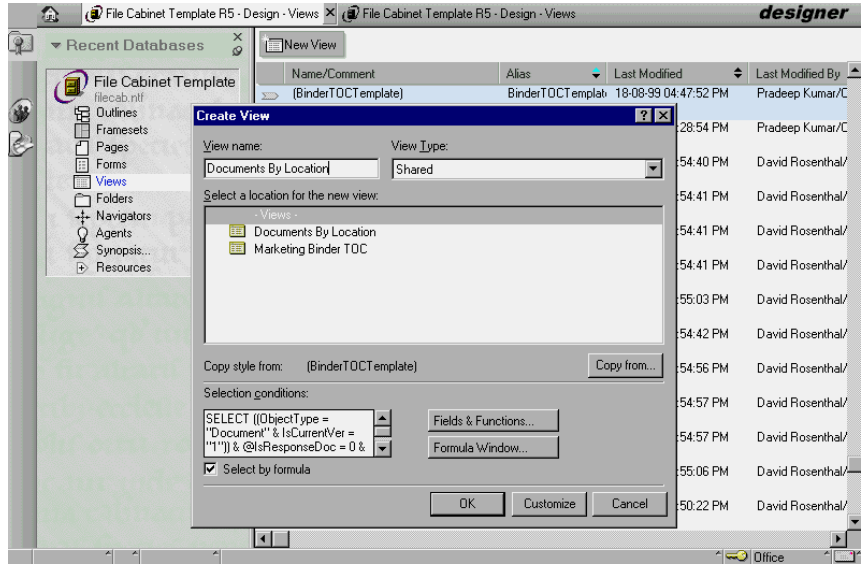
Caution Do not alter the columns to the left of the Document Title column.

Note If you modify this view at a later date, you should rebuild the folder for any binder that uses this view. We discuss rebuilding folders in a later chapter of this book.

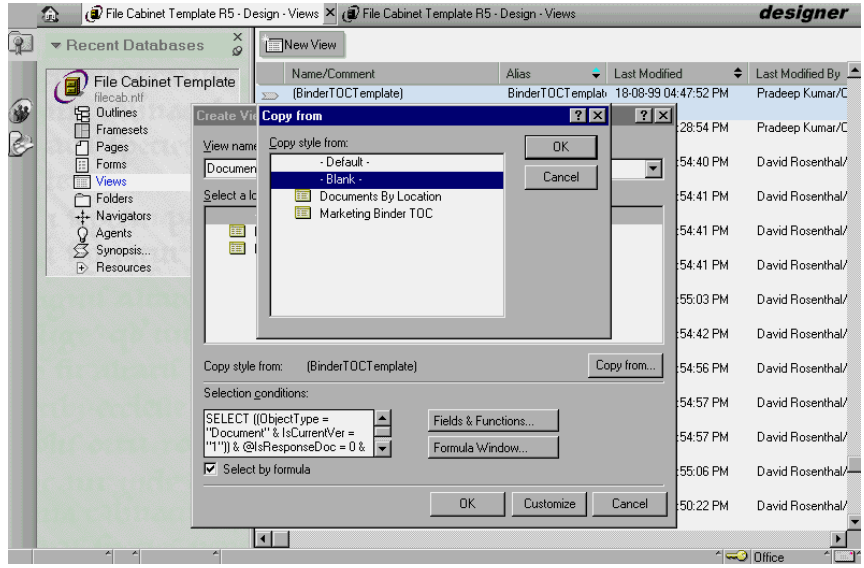
To create a categorized view for travel locations, follow these steps:

1. Select the File Cabinet Template database and choose Create - Design - View. The Create View dialog box is displayed.
2. Enter the view name as Documents By Location.

3. Select the view type Shared.



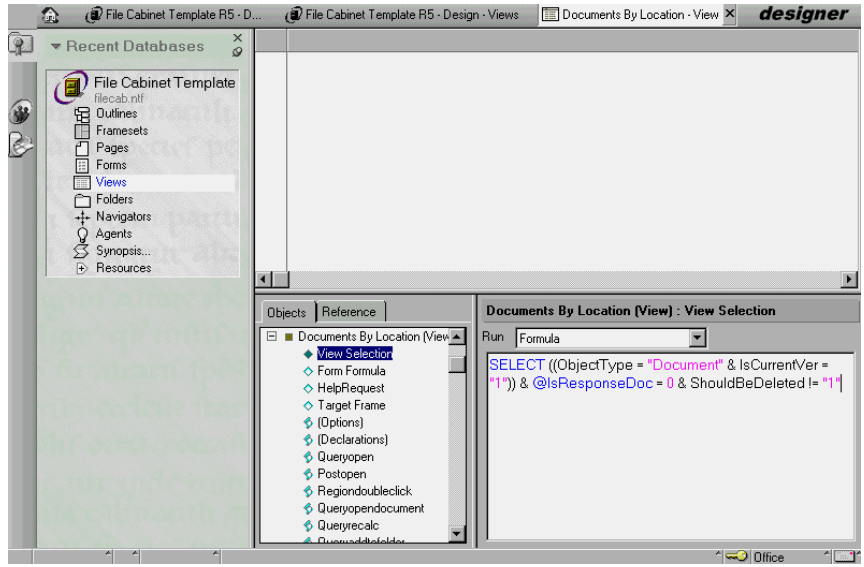
4. Click Copy from and select -Blank-.



5. Click OK.
6. A new view called Documents By Location appears in the view list.
7. Select this new view and double-click to open it.

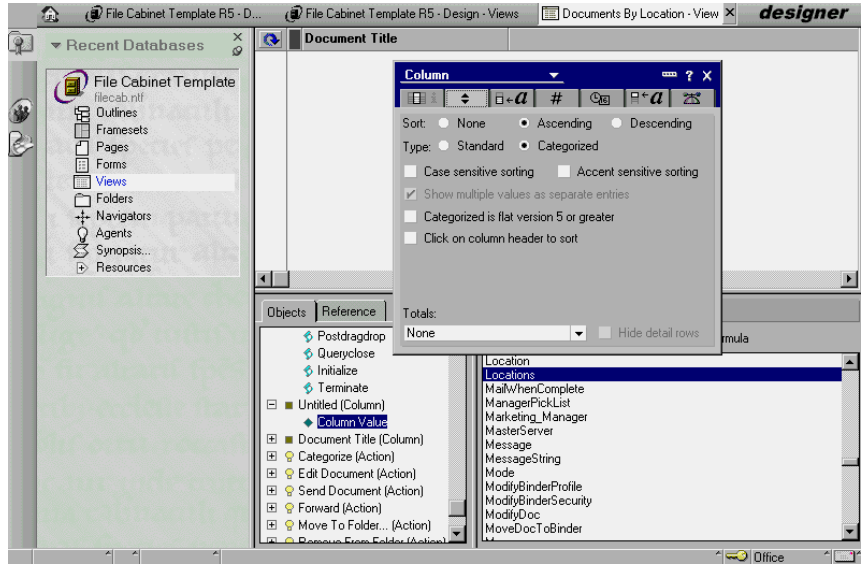
8. For view selection, type in the formula:

```
SELECT ((ObjectType = "Document" & IsCurrentVer = "1")) &
@IsResponseDoc = 0 & ShouldBeDeleted != "1"
```



9. Choose Create - Insert New Column.
10. For Column value, select the field in the Programmers' pane and mark the field Locations.
11. Select the new column and choose Design - Column Properties.
12. Leave the Title field under the Column info tab blank.

13. Under the Sorting tab (tab number two), select Categorized as Type.



14. Choose Create - Append New Column.
15. For Column value, select the field Title.
16. Select the new column and choose Design - Column Properties.
17. Enter Document Title as column title.
18. Save and close the view.

This view will display document titles categorized according to the locations where a document is attached.

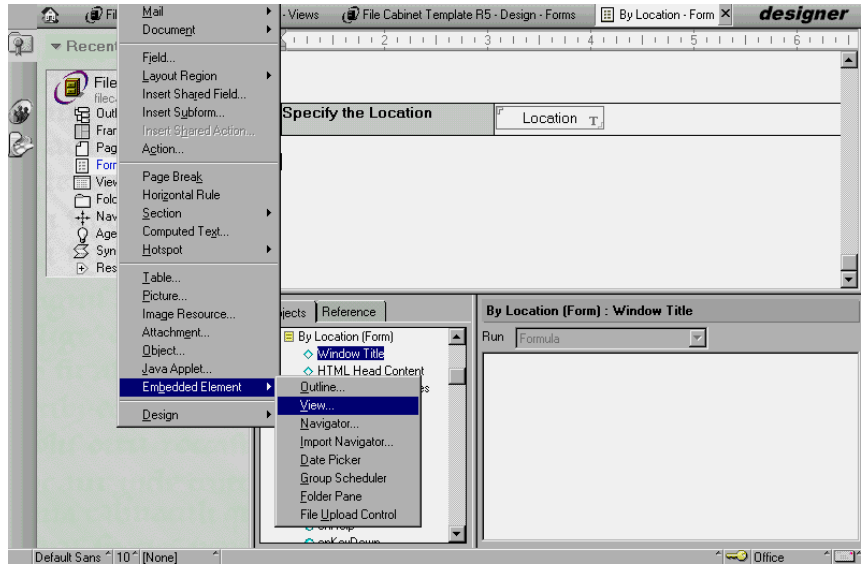
Single category view

Single category view is a new feature available in Domino R5.0. Using this, you can display a subset of documents based on a specified category value.

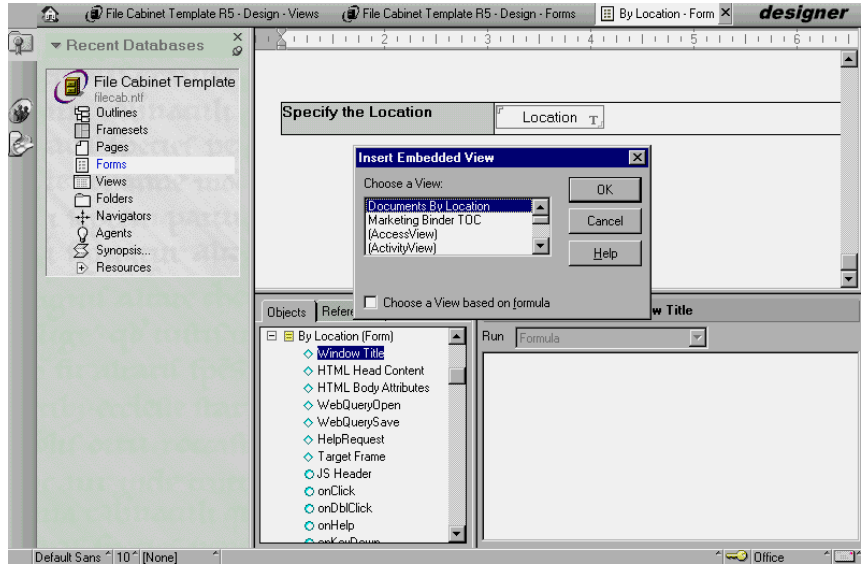
Create a single category view to display documents based on a location specified by the user. When the user enters a location name in the location field, all documents related to that location will be displayed.

To create a single category view, perform the following steps:

1. Select the File Cabinet Template database and choose Create - Design - Form.
2. Create a field called Location.
3. Choose Create - Embedded Element - View. The Insert Embedded View dialog box is displayed.

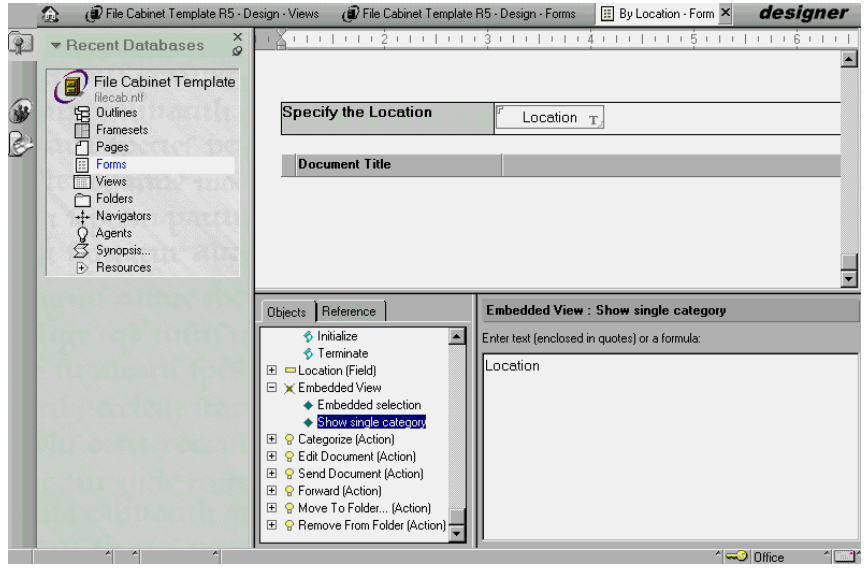


4. Select the Documents By Location view and click OK.



5. Select Show single category under Embedded view object.

6. Type Location for the formula.



7. Choose Design - Form Properties.
8. Name the form By Location.
9. Save and close the form.

Note After modifying the design of a template database, you should refresh the designs of databases created with that template for the changes to become effective. To refresh the database design, choose File - Database - Refresh Design or type the following on the server console command line:

```
load design
```

Summary

In this chapter, we discussed modifying a Domino view to customize Domino.Doc solutions. We discussed modifying existing views, and the creation of new views.

Chapter 7
Customizing navigators

This chapter describes what navigators are, where they are used in Domino.Doc library and File Cabinets, how they look, and how you can customize them.

It also explains how Web users navigate through Domino.Doc.

By the end of this chapter, we will have walked through how to create a new navigators to achieve Millennia's business requirements.

What navigators are

Database navigators are graphical interfaces that allow users to easily access views, Domino data, or other applications. Navigators can include graphic buttons or hotspots, which are programmed areas users can click to execute an action.

Where navigators are used in Domino.Doc

Every single type of database in Domino.Doc has its own set of navigators, but we are going to concentrate on customizing the library and file cabinets because they are the databases that most of the users will be using.

You can see how the navigators in the library and file cabinet template look in Appendix C: Domino.Doc navigators.

How Web users see Domino.Doc navigators

All navigators described in the Appendix C: Domino.Doc navigators are designed for users that access Domino.Doc via a Notes Client.

Notes forms, views, and documents containing HTML and Java Script code are specially created to provide Web users with access to Domino.Doc.

The navigators shown to Web users are generally forms with names that begin with "Nav-".

Millennia's requirements for modifying navigators

Millennia Space Travel Corporation's navigational requirements and reasons for them are the following:

Requirements	*Reasons*
Remove unused buttons	To avoid having users see or perform actions that they are not authorized to do.
Include Millennia's logo	It is a company business rule.
Modify the navigator colors and buttons	To make them look different and minimize the scroll bar usage.
Create a full screen navigator	To have a starting point.
Use new R5 navigation features	To take advantage of R5 features .

Our discussion of customizing Millennia's navigator is divided into the following three categories to make the concepts easier to understand:

- Modifying existing navigators
- Creating and using a new navigator
- Using new R5 design elements instead of navigators

Modifying existing navigators

We will modify existing navigators to remove unused buttons, include Millennia's logo, and change existing colors and buttons.

Removing unused buttons

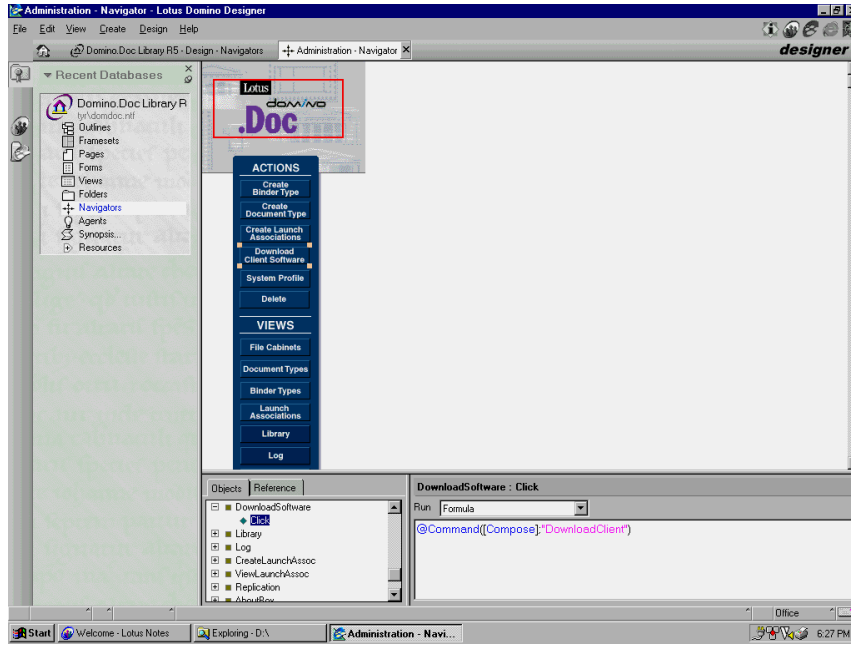
Millennia's users should not be able to download or install the Domino.Doc Desktop Enabler because Millennia uses IBM Tivoli to distribute and manage their client software files.

For the same reason, Millennia will need to remove the Getting Started button from the navigators.

Modifying the library template

Use the following steps to modify the library template navigators to achieve Millennia's goals:

1. Open the Library Template (domdoc.ntf) in your Domino Designer client.
2. Open the Administration navigator.



3. Click the Download Client Software graphic, then press the Delete key.
4. Click the System Profile action.
5. Drag the action up to align it with the others.
6. Repeat step 5 to align the other actions.
7. Choose File - Save.
8. Choose File - Close.

After modifying the navigators, delete the DownloadSoftware field on the Nav-Admin form, because Web users access this form instead of a navigator.

Note You will need to wait until the server task Designer runs for the changes to take effect on the libraries, or you can update the database design manually.

Modifying the file cabinet template

Use the following steps to modify the file cabinet template navigators to achieve Millennia's goals:

1. Open the File Cabinet Template (filecab.ntf) in your Domino Designer client.
2. Open the (DocumentNavNext) navigator.
3. Click the Getting Started action, then press the Delete key.

4. Click the Edit File Cabinet action.
5. Drag the action up to align it with the others.
6. Repeat step 5 to align the other actions.
7. Choose File - Save.
8. Choose File - Close.

Use the same procedure to customize the following navigators:

- (DocumentNavPrevNext)
- (DocumentNavPrev)
- (DocumentNav)
- (HomeNavforAllbyAuthorView)
- (HomeNavforAllbyTypeView)
- (HomeNav)

After modifying the navigators, delete the GettingStarted field on the following forms that are used as navigators for Web users:

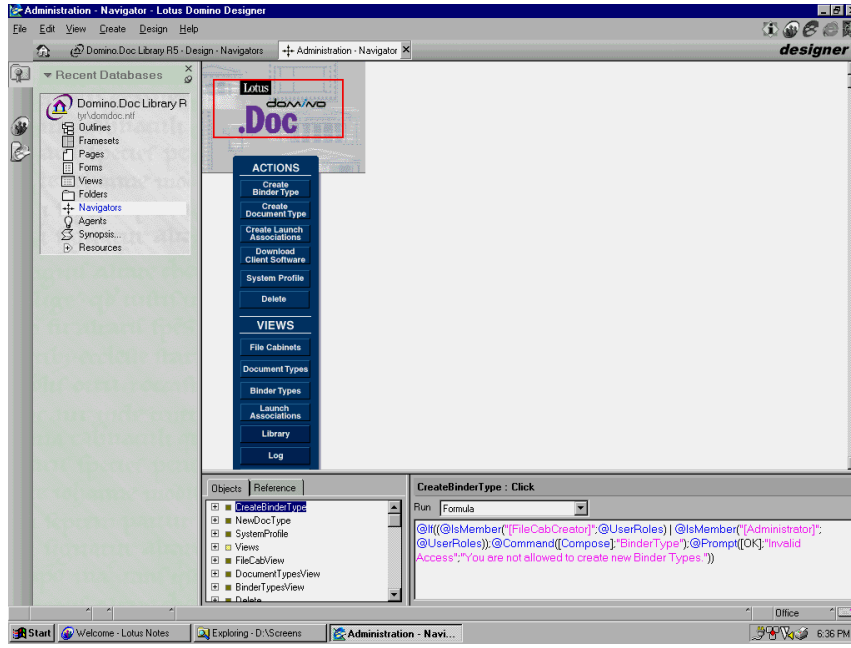
- Nav-CustomViews
- Nav-DocumentViews
- Nav-Main

Note You will need to wait until the server task Designer runs for the changes to take effect on the file cabinets, or you can update the database design manually.

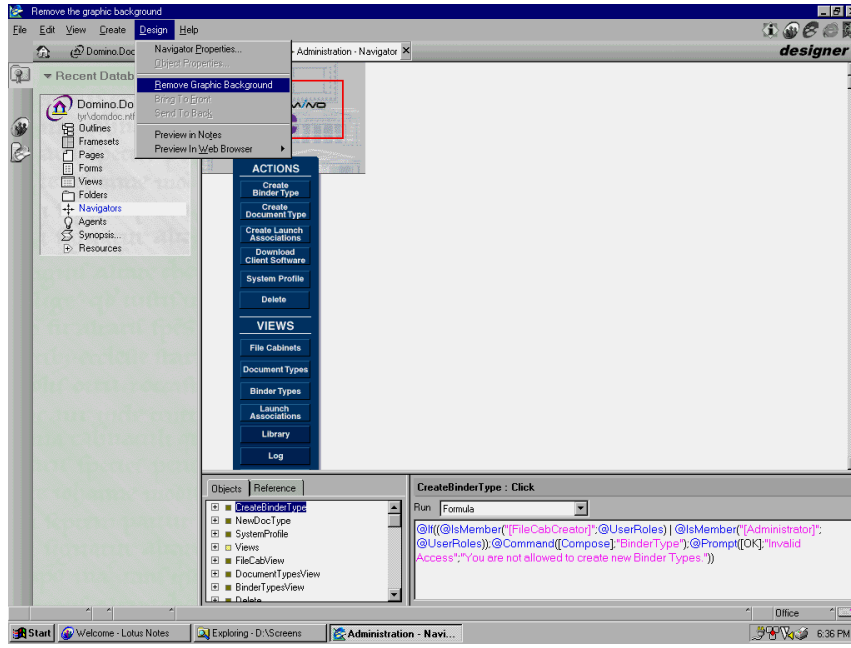
Inserting Millennia's logo in the existing navigators

Follow these steps to insert Millennia's logo in the navigators:

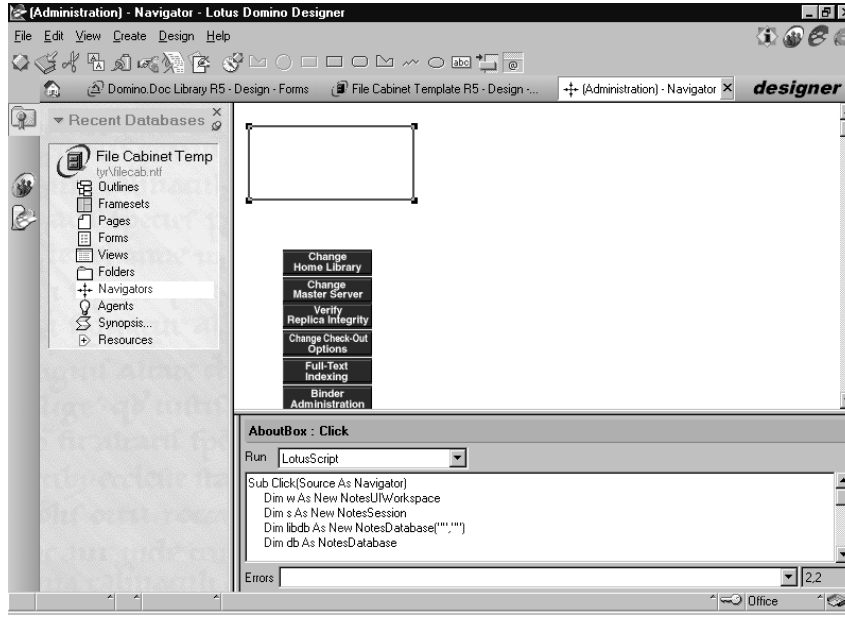
1. Open the Library Template in your Domino Designer client.
2. Open the Administration navigator.



3. Choose Design - Remove Graphic Background.



4. Click the remaining hotspot.



It gives users the opportunity to see the Domino.Doc Help About dialog box. If you want your users to see this document, perform the following steps to unlock its position. Using the object properties, move it to the desired destination, and lock it there.

a. Click the remaining hotspot, then click the Properties icon.

b. On the Information tab, deselect the Lock size and position check box.

c. Drag the hot spot to the new desired destination.

d. Resize it as needed.

e. On the Information tab, check the Lock size and position option.

We will not keep this hotspot for the Millennia navigator. Press Delete to remove it.

5. Locate the laction.gif file located in the Domino icons directory. This file has the graphic background we just deleted from the navigator.

If you have a third-party tool like PaintShop Pro you can edit it as needed, and then copy it to the clipboard.

If you do not have any tools that work with gif files, you can copy it to the clipboard by using the Notes client.

a. Open your mail.

b. Create a new memo. Put the cursor in the body text field.

c. Select File - Import.

d. Notes shows a file dialog box. Locate the laction.gif file and click the Import button.

e. When the file has been imported, make sure it still has the selection focus. If not, click it with the mouse and then select Edit - Copy.

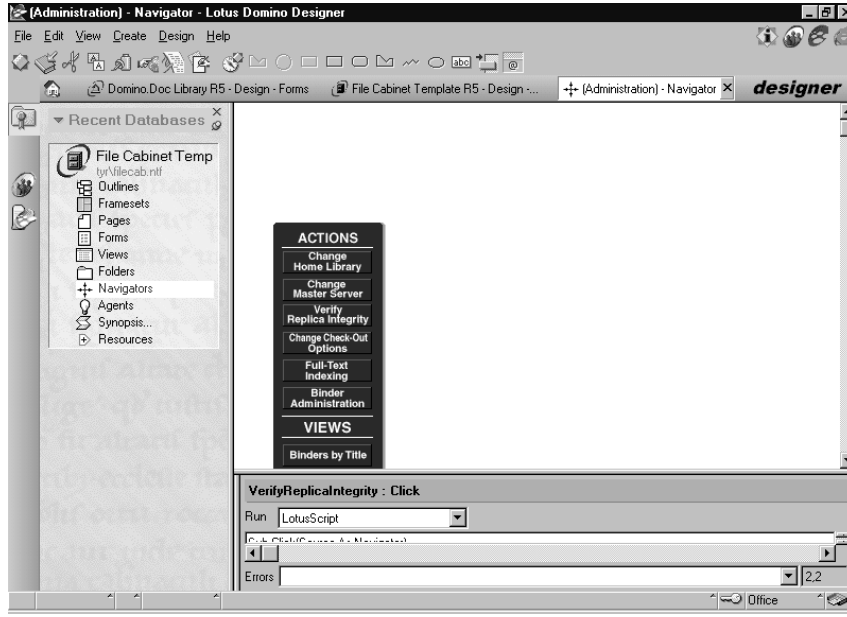
f. The laction.gif file has now been copied to the clipboard and you can discard the new memo you just created.

If you do not have a Notes client, you can open the laction.gif file in Microsoft Internet Explorer. Right-click and select Copy. This also copies the file to the clipboard.

Important This gif file is used by Web users. Saving the changes you made will affect their navigators' appearance too.

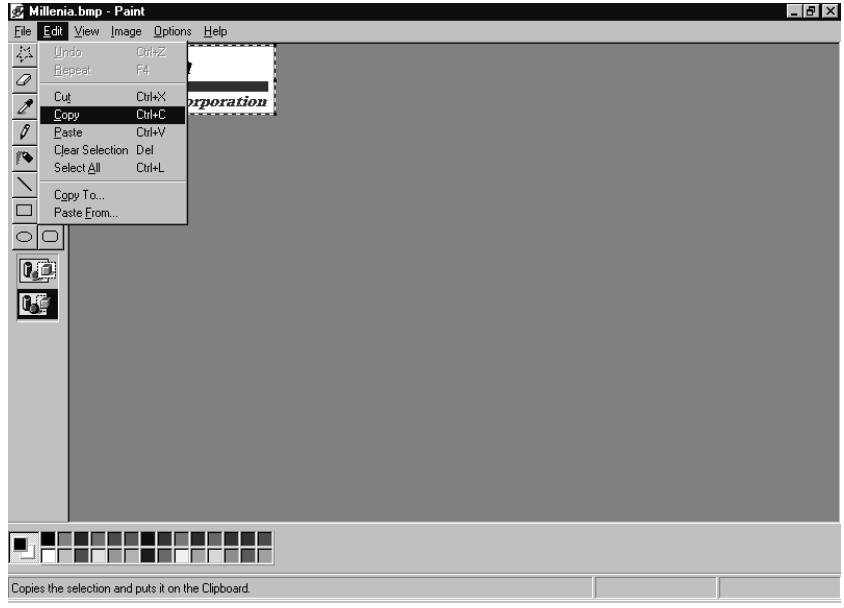
6. Switch to the Domino.Doc Administration navigator window.

7. Choose Create - Graphic Background.



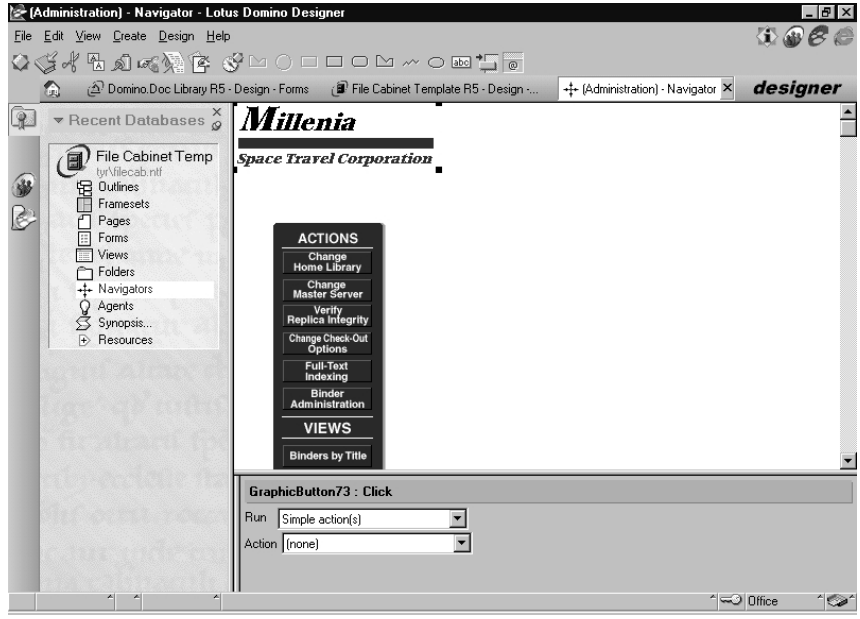
8. Open the company logo using the MS Paint tool.

9. Select the logo, and then select Edit - Copy.



10. Switch to the Domino.Doc Administration navigator window.

11. Choose Edit - Paste, then drag the logo to the desired Location.



12. Choose File - Save.
13. Choose File - Close.

Repeat these steps for all navigators in this database, and for all navigators in the file cabinet template database.

Note The steps above describe modifying the navigator as Notes users see it. To display the Millennia logo for Web users, you must modify the file homenav.gif in the domino/icons directory under the Domino data directory.

Modifying navigator colors and buttons

The goal for modifying the navigator colors and buttons is that, Millennia users will not need to scroll the navigator screen to click a button, and if they need to print the screen for any reason, they can print the whole navigator as well.

Follow these steps to modify the Millennia navigator color and buttons:

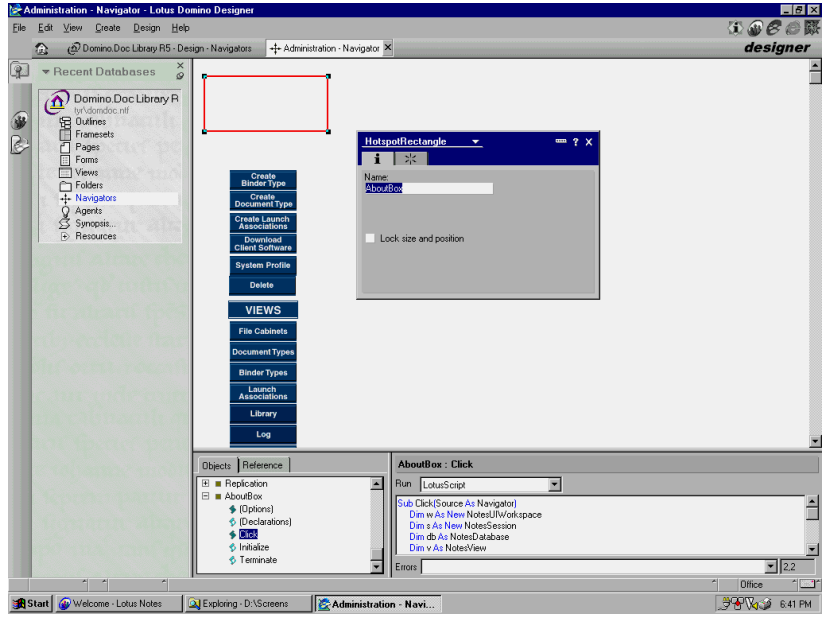
1. Open the Library Template in your Domino Designer client.
2. Open the Administration navigator.
3. Choose Design - Remove Graphic Background.

4. Click the remaining hotspot.

   This hotspot gives users the opportunity to see the Domino.Doc Help About dialog box. If you want users to be able to see this document, the following steps will unlock its position using the object properties, move it to the desired destination, and lock it there.

   a. Click the remaining hotspot, then select Design - Object Properties from the menu.

   b. On the Information tab deselect the Lock size and position check box.



   c. Drag the hotspot to the desired destination.

   d. Resize it as needed.

   e. On the Information tab, select the Lock size and position option again.

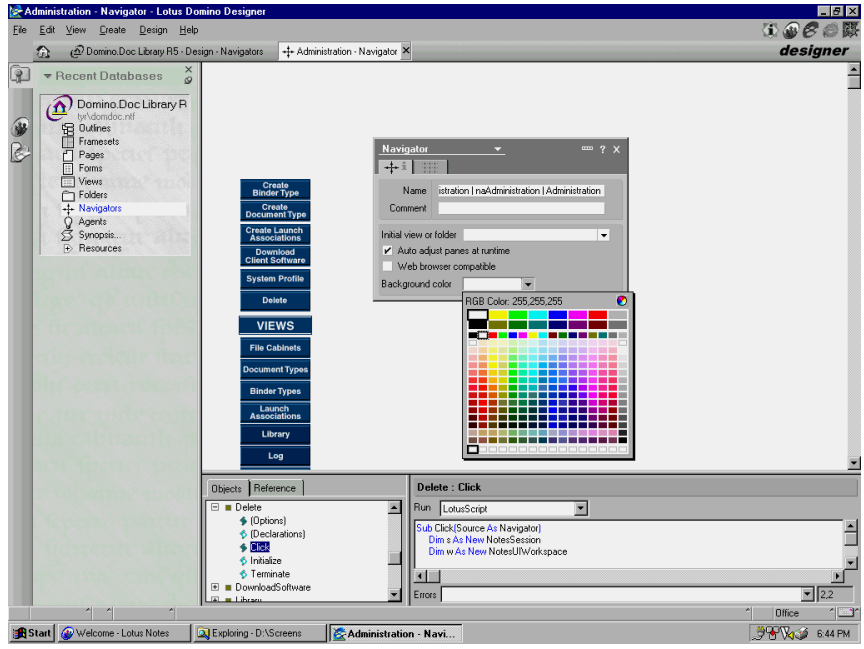
   f. Close the properties dialog box.

   In our example, we will simply remove the hotspot by pressing Delete.

5. Rename or delete the laction.gif file located in the Domino icons directory.

   Important This gif file is used by Web users. When you rename or delete it, you make the navigator background unavailable to these users. If you want your company logo to appear to Web users, the easiest way to do this is by renaming the company logo file to laction.gif or homenav.gif and copying it into the Domino icons directory.

6. Select Design - Navigator Properties from the menu.
7. Choose one color for the Background Color Field from the Navigator Info tab.

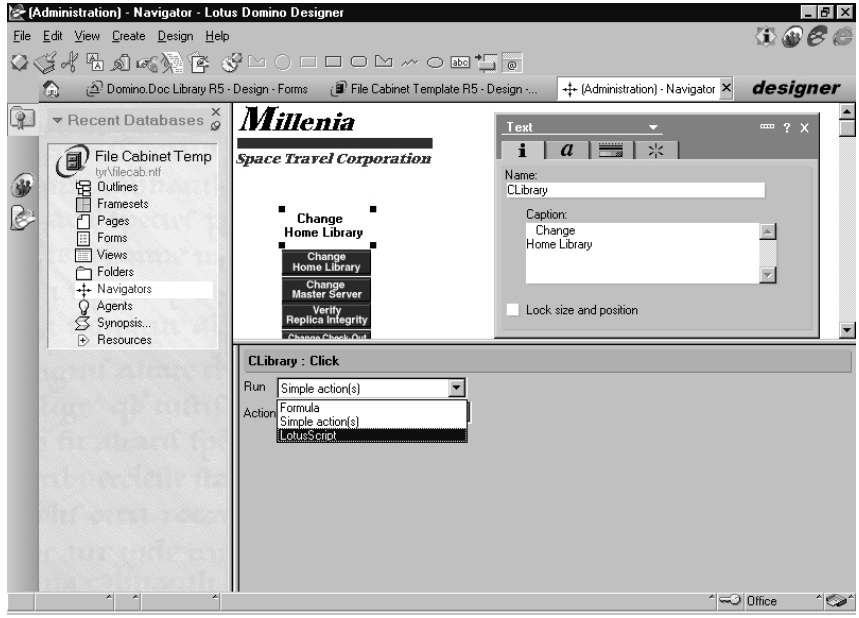


8. Close the Properties dialog box.
9. Click the first action and copy its code to the clipboard by selecting it with the mouse, then choosing Edit - Copy.
10. Create a text rectangle over the first action and enter text for the action. In our example, we will use the same text as for the action we are replacing: Change Home Library.

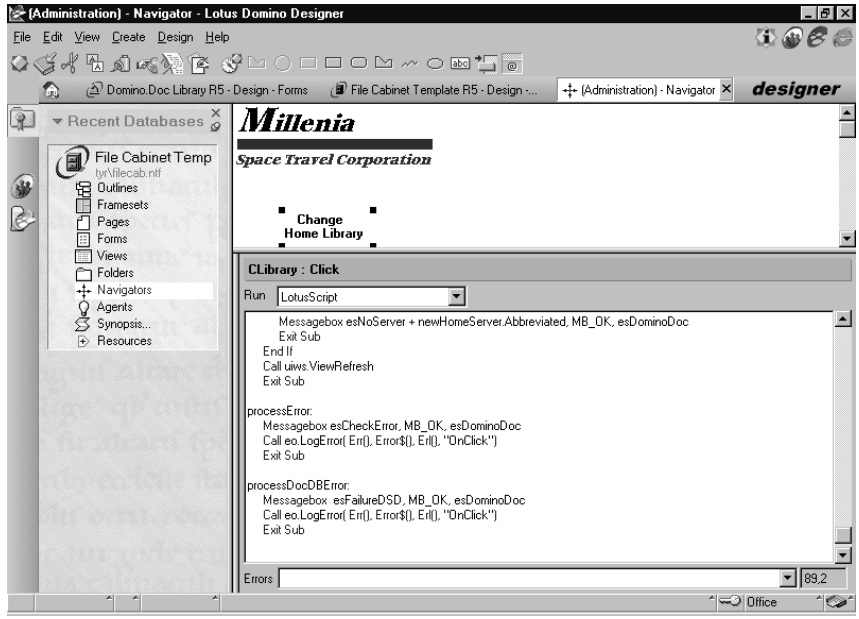
Note Instead of using a text rectangle, you could also use a text button, a graphic button, or a pasted image for your new action.

11. Create a hotspot action and place it around the text you have just entered.

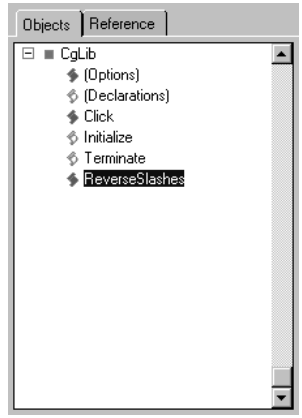
12. Select LotusScript in the Run field on the Programmers pane.



13. Place the cursor in the entry area of the Programmers pane, then choose Edit - Paste.

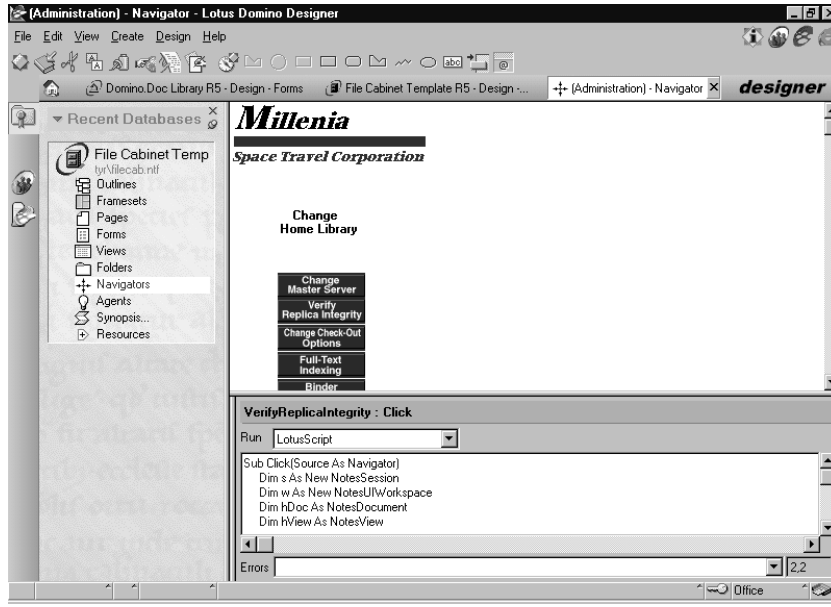


Important In this example, you will also need to copy the (Options) event code and copy the Reverse Slashes function code. You do this by selecting the original action. Then you select Options in the Info list (shown below) in the Programmers pane. This displays the (Options) code in the Programmers pane. You copy the code to the clipboard as described above, then select your new action and (Options) in the Info list box before pasting in the code. You should repeat these steps for the ReverseSlashes code.



Important Be sure you have copied the code related to all these events and functions before proceeding with the next step.

14. Delete the original graphic action button by selecting it, and then pressing the Delete key.



15. Choose File - Save.
16. Choose File - Close.

Repeat these steps for actions in this navigator and all other navigators in this database, and the file cabinet template navigators, too.

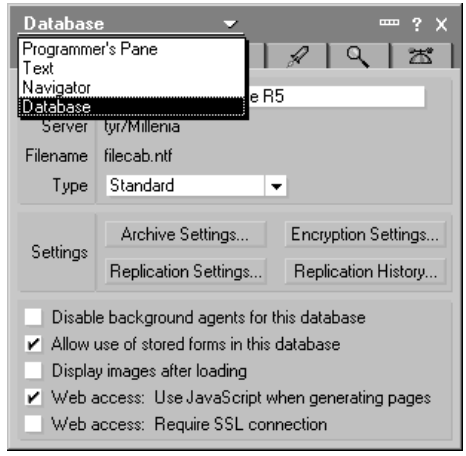
Note Since Web users use forms as navigators, you will need to change fields within the "Nav-" forms to change their navigators' look and feel.

Creating and using a new navigator

Millennia would like to provide a new and easy starting point for their Domino.Doc users.

Follow these steps to create a new navigator:

1. Open the Library Template in the Domino Designer client.
2. Choose Create - Design - Navigator.
3. Include some actions and pictures as desired.
4. Choose File - Save.
5. Choose File - Close.
6. Click the Properties icon.
7. Select Database as shown in the figure below.

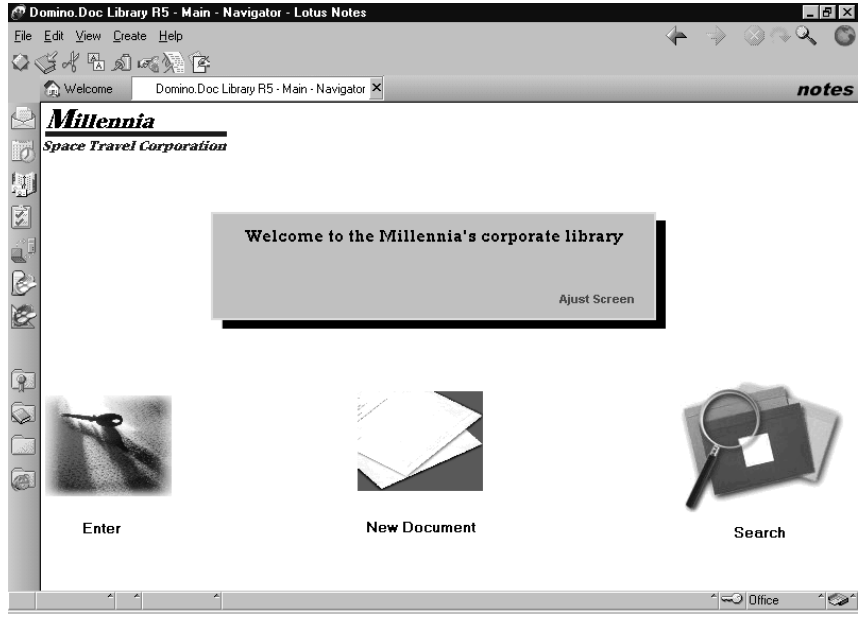


8. Click the Launch tab (tab number five). Here you can decide what Domino should do when the database is opened by a user. Among the choices for "When opened in the Notes client," choose Open designated Navigator.

   Domino displays a drop-down list for navigator type, and one with the names of navigators.

9. In the Type of Navigator field, select Standard Navigator.
10. In the Name field, select the name of the navigator you have just created.
11. Close the Properties dialog box.

The following figure shows Millennia's main navigator:



In this navigator, the Millennia logo and the library banner are pasted in as pictures. In the library banner there is a hotspot that a user, using a 640X480 screen monitor, can click to adjust the navigator for their screen.

This "Adjust Screen" action actually switches the display to a different navigator that displays better on machines running a 640X480 screen configuration. This hotspot has the following code:

```
@Command([OpenNavigator];"Main640X480")
```

The three graphics at the bottom of the screen are also hotspots with formulas associated with them.

The Enter hotspot graphic of a Key navigates the users to the library and has the following formula:

```
@Command([OpenNavigator];"Library");

@Command([OpenView];"Main View")
```

The New Document hotspot graphic provides an opportunity to create a new document within the library with just one click, and has the following formula:

```
FCList := @DbColumn("" ; "" : "" ; "(ExternalCabinetLookup)" ;
1);

FileCabinet := @Prompt([OKCANCELLIST] ; "Select a File Cabinet"
; "" ; @Subset(FCList ; 1) ; FCList);

@If(FileCabinet = "" ; @Return("") ; "");


RepID := @DbLookup("" ; LibServer : LibDb ;
"(ExternalCabinetLookup)" ; FileCabinet ; "BinderDbReplicaID");

filecab := @Left(RepID ; 8) + ":" + @Right(RepID ; 8);


DSReplica := @DbLookup("" ; filecab ; "(InternalAdmin)" ;
"GlobalProfile" ; "DefaultDocReplica");

DSRepID := @Left(DSReplica ; 8) + ":" + @Right(DSReplica ; 8);


@Command([FileOpenDBRepID]; DSRepID; ""; "(All Documents)");

@Command([Compose]; ""; "NewDocument")
```

The Search hotspot with a magnifying glass graphic opens the Search form. By clicking this button, users can start searching for a single document in the whole library. This button has the following formula:

```
@Command([Compose];"NotesSearch")
```

Using new R5 Design elements instead of navigators

Framesets, pages, and outlines are new Design elements in Domino R5 that help you to create a structured screen.

Framesets provide a multi-pane interface for the user.

Pages are special easy-to-use forms that can be used by application developers who are not familiar with traditional Domino development, but who are familiar with Web development tools. A page can be easily designed in Domino Designer in a manner similar to how a Web page is designed with products such as MacroMedia DreamWeaver. They are predominantly used to display text, images, and applets.

Note Pages cannot contain fields, layout regions, actions, subforms, or some embedded objects.

Note Pages can be launched when the database is opened, while forms cannot.

Outlines are similar to site maps. Outlines enable you to create a hierarchical structure of links and elements for your site.

Resources

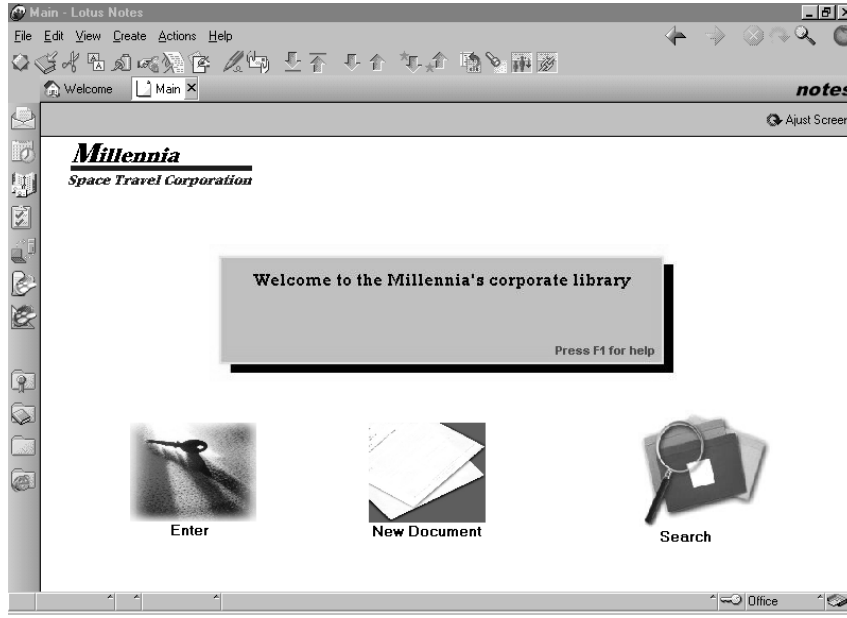
There are five different types of resources:

- Images
- Applets
- Shared Fields
- Script Libraries
- Others

The advantage of using these resources is that you can store one of these elements for later reuse. For example, when you want to use your company logo in several different places, you can create the company logo and store it as an image resource, and then insert the logo wherever you want by choosing Create - Image Resource from within Domino Designer.

This enables you to control and manage your applications more easily.

The following figure is an example of how to use R5.0 Pages and Image resources to create a starting point for Domino.Doc for both Notes and Web users.



On this page, all the images are actually Image resources with action hotspots, and we moved the code to adjust the screen into an action at the upper right corner of the screen. It works almost the same way as the one used in the navigator; the only difference is that it opens another page instead of a navigator.

Note This page will be accessed only by Millennia's Notes users. However, you can also design a page for your Web users. If you want to do this, you must designate a new database to hold the page and be the user's starting point. You cannot place the page in the library database because it launches the first document in a view called (HTMLLaunch) when it is opened via a Web browser.

This "Adjust Screen" action in reality is only a page change to one that displays well on machines running a 640X480 screen configuration. This hotspot has the following code:

```
@Command([OpenPage];"Main640X480")
```

The three images at the bottom of the screen are image resources with hotspot actions and associated formulas.

The Key image takes the library users to the library; this hotspot has the following formula:

```
@PostedCommand([OpenNavigator];"Library");

@Command([OpenView];"Main View")
```

Note This code is slightly different from the code used in the navigator construction. The main reason for this is to avoid the @command's side effect (it does not work as expected).

The Document image provides users with the opportunity to create new documents within the library with just one click. This hotspot has the following formula:

```
FCList := @DbColumn("" ; "" : "" ; "(ExternalCabinetLookup)" ;
1);

FileCabinet := @Prompt([OKCANCELLIST] ; "Select a File Cabinet"
; "" ; @Subset(FCList ; 1) ; FCList);

@If(FileCabinet = "" ; @Return("") ; "");


RepID := @DbLookup("" ; LibServer : LibDb ;
"(ExternalCabinetLookup)" ; FileCabinet ; "BinderDbReplicaID");

filecab := @Left(RepID ; 8) + ":" + @Right(RepID ; 8);


DSReplica := @DbLookup("" ; filecab ; "(InternalAdmin)" ;
"GlobalProfile" ; "DefaultDocReplica");

DSRepID := @Left(DSReplica ; 8) + ":" + @Right(DSReplica ; 8);


@Command([FileOpenDBRepID]; DSRepID; ""; "(All Documents)");

@Command([Compose]; ""; "NewDocument")
```

Note This code is exactly the same as the one used in the navigator construction.

The Magnifying glass image opens the search form. By clicking this icon, users can start searching for a single document in the whole of the library. This hotspot has the following formula:

```
@Command([Compose];"NotesSearch")
```

Note This code is exactly the same as the one used in the navigator construction.

Please see *Domino Designer Help* for details on creating framesets, pages, outlines, and resources.

Summary

In this chapter, we discussed Domino.Doc navigators, what navigators are available in the template files, and how to customize these navigators for the Millennia application.

At the end of this chapter, we discussed creating new navigators to have a single point of entry for the Millennia library.

Chapter 8
Document life cycle

Domino.Doc manages documents throughout their life cycle — from authoring through review, approval, distribution and archiving. With document life cycle management, Domino.Doc helps organizations to manage their documents predictably and efficiently.

In this chapter, we discuss what document life cycle is and how Domino.Doc handles the review and approval process within the document life cycle. We also describe how to initiate the review and approval process and how to set up a defined review and approval cycle so that all documents are examined by the appropriate reviewers and approvers.

What the document life cycle is

Every document in an organization goes through a set of processing stages or life cycle events, from creation through archiving. These life cycle events include authoring, review, approval, distribution, and archiving. The number of processing stages and the nature of processing may differ from document to document, depending on each document's importance. It is very important for the success of an organization to efficiently manage documents throughout these processing stages. Thus, organizations would like to have a document management solution which will take care of document life cycle. They may want to set up a predefined life cycle so that all documents created will surely go through this cycle and the management will have full control over the documents. One of the important features of Domino.Doc is that it provides life cycle management for documents.

In this section, we will discuss what a document's life cycle events are, and how Domino.Doc can manage these events.

Authoring

This is the stage when somebody is actually working on the document. The document in this stage is just a draft, not the final version. Other workers should not be able to work on the same document, to prevent one person making changes that are overwritten by another person. Each worker should have full control over the documents they are working on.

By means of check in and check out control, Domino.Doc ensures that any document currently being worked on by one person cannot be modified by another.

Review and Approval

In this stage, the document is reviewed and approved by one or more people. Domino.Doc allows you to define a review and approval cycle for each document type so that each document will pass through a predefined set of reviewers and approvers.

Distribution

In this stage, documents are published so that they are accessible to other users. Any document in this stage is the final, approved version. Domino.Doc, by integration with Domino Workflow and other applications, facilitates the distribution of documents across the organization.

Archiving

Once the useful life of a document is over, it should be archived for future use. Domino.Doc, in conjunction with Domino.Doc Storage Manager, provides an easy and efficient archiving and retrieval method.

In the rest of this chapter we will concentrate on how Domino.Doc supports the management of review and approval.

Review and approval cycles

The review and approval cycle for documents is very important in managing collaborative documents, as this helps management to have full control over the documents. There are many business circumstances in which documents require review and approval before they can be finalized or published. For instance, a marketing presentation might require review by technical experts and salespeople; finance documents prepared by accountants may also be reviewed by editors; work rule policy documents may need approval by the human resources department, legal counsel and the union representative before they are distributed to end users. Using Domino.Doc, you can set up a review and approval cycle so that documents will be reviewed and approved by the correct people in an organization before they are published to the users. The individuals or groups who should be in the review and

approval cycle can be specified at the same time the document type is defined, or draft editors can determine who should review and approve particular documents when they submit each document for review and approval.

Review cycle

Many organizations require their documents to be reviewed, commented on and edited by a group of people before publishing.

Any draft editor can submit the document draft for review, provided drafts are not disabled for the document type and a review cycle is set up for the document type. The document will be routed to different reviewers either serially or in parallel, depending on how the review cycle was set up. The document status will change to "In Review." Once the last reviewer finishes his review, the initiator is notified and the document status changes to "Review Complete." The initiator can change the reviewer list or cancel the review cycle at any time prior to review completion.

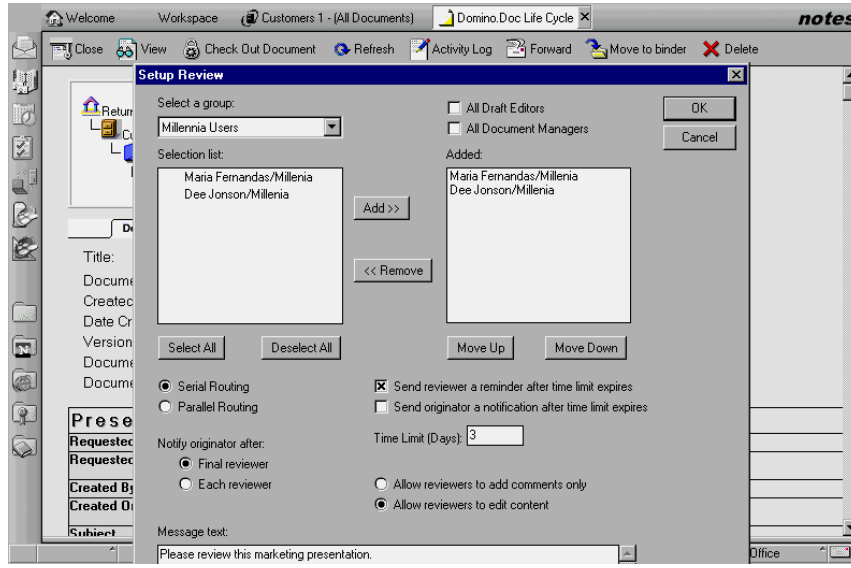
Initiating review cycle

Only document drafts can be submitted for review. If the document you want to review is currently a "version," you should check it out first and then check it in as a draft.

To initiate the review cycle, perform the following steps:

1. Open the library by double-clicking the database icon.
2. Open the file cabinet and the binder that contains the document you want to submit for review.
3. Open the document profile and click Submit for Review. The Setup Review dialog box is displayed. The Setup Review dialog box will have the values specified in the document type as default values. The draft editor can overwrite these values.

Note If you are working in a non-master file cabinet, you should first check out the current draft.



4. Specify the document reviewers.
   - Select “All draft editors” to make all draft editors reviewers.
   - Select “All document managers” to make all document managers reviewers.
   - Select “Specify individual reviewers” to make specific individuals reviewers and specify their identities.

   To select individuals as reviewers, follow these steps:

   a. Select a group from the list.

   b. The names of group members are displayed in the selection list.

   c. Click Select All if you want all members to be in the reviewers list.

   d. Select individual names and click Add to add each specified name to the reviewers list.

   e. Click Deselect All if you want to clear your selection.

   f. Use the Move Up and Move Down buttons to arrange the members in the order you want them to review the document. This is important if you select the Serial routing type since the document will be routed to one at a time, in the order they are arranged in the list.

Note It is also possible that users were invited to participate in the file cabinet as individuals, in which case their names will appear in the Selection list automatically, and no groups will need to be selected. However, it is recommended to work with groups in the file cabinets as this normally requires less administrative work.

1. Specify the routing type.
   - Choose "Serial" if you want reviewers to review the document one at a time, in the order you specified in the list.
   - Choose "Parallel" if you want each reviewer to review the document at the same time. (That is, review the document without seeing any other reviewers' comments or edits.)
2. Specify the time limit option.
   - Select "Send reviewer a reminder after the time limit expires" if you want to notify the reviewer after a specified time.
   - Select "Send originator a reminder after the time limit expires" if you want to notify the originator after a specified time.
3. Specify whether to notify the originator after each review or after final review.
4. Specify whether the reviewers should be able to modify the document or only to add comments.
5. Enter the message to be sent to reviewers.
6. Click Save if you are working in a browser.
7. Click OK if you are working in Domino.

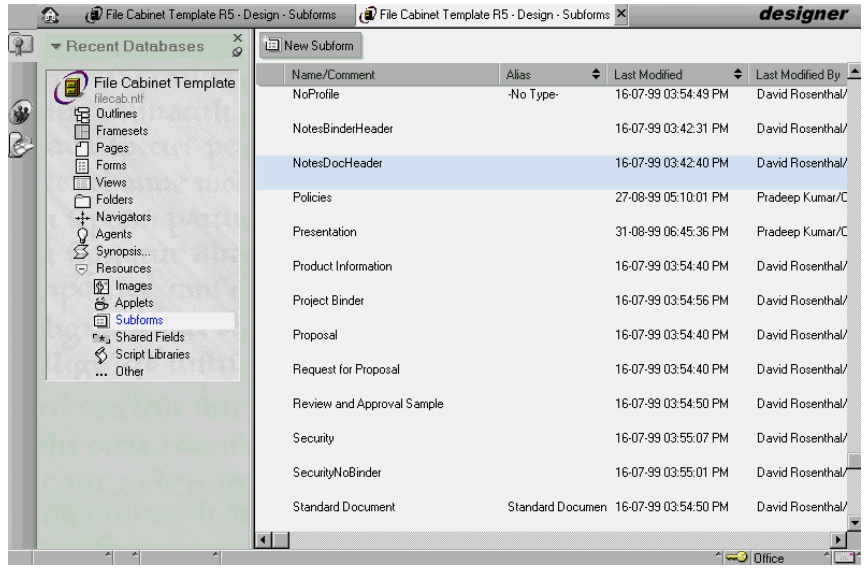
The reviewers will be notified by e-mail that they have documents to review. The e-mail will contain a link to the document which the reviewer can access by clicking on the link. The reviewer can edit the document or add comments to the document as per the setting described previously.

Predefined review cycle

In the previous section, we described a review cycle in which the draft editors can set up the review cycle. The draft editors can overwrite the default values specified in the document type. But in some cases, organizations require a predefined review cycle and all the documents are required to follow the review cycle defined in the document type. The draft editors should not be able to change this setup.

To hide the Setup Review dialog box from the draft editors while they are submitting the documents for review, you should modify the design of the file cabinet template as follows:

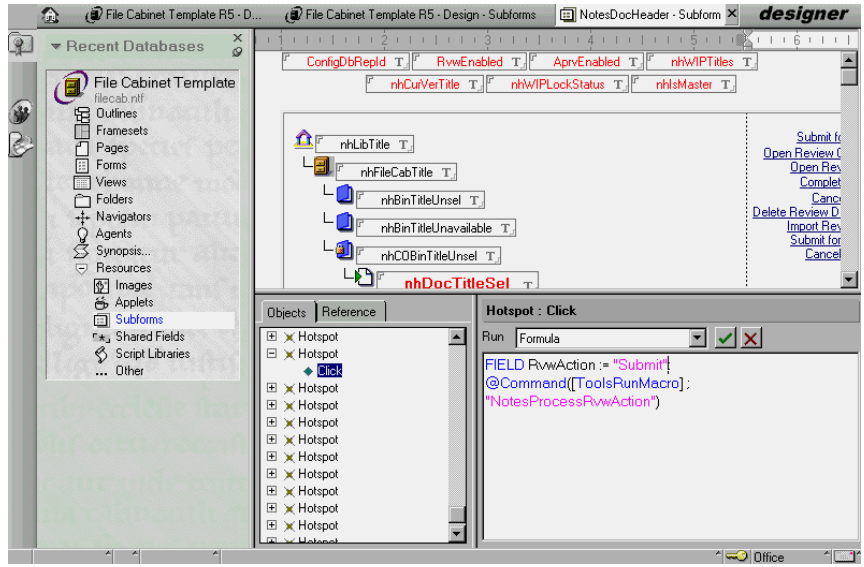
1. Open the File Cabinet Template (filecab.ntf) in Domino Designer.
2. From the design elements list, select Resources - Subforms. The list of subforms in the template is displayed.



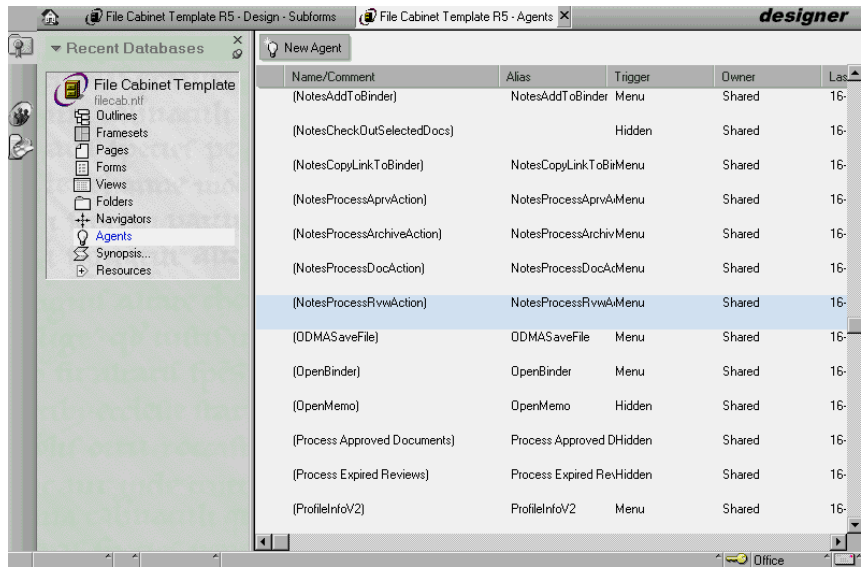
3. From the list, select NotesDocHeader and double-click to open it.
4. Click the hotspot Submit for Review. The formula for the hotspot will appear in the formula window.
5. Replace the existing formula with a new formula as follows:

```
FIELD RvwAction := "Submit";

@Command([ToolsRunMacro] ; "NotesProcessRvwAction")
```



6. Save and close the subform.
7. Select Agents from the design elements list. The list of agents in the template is displayed.

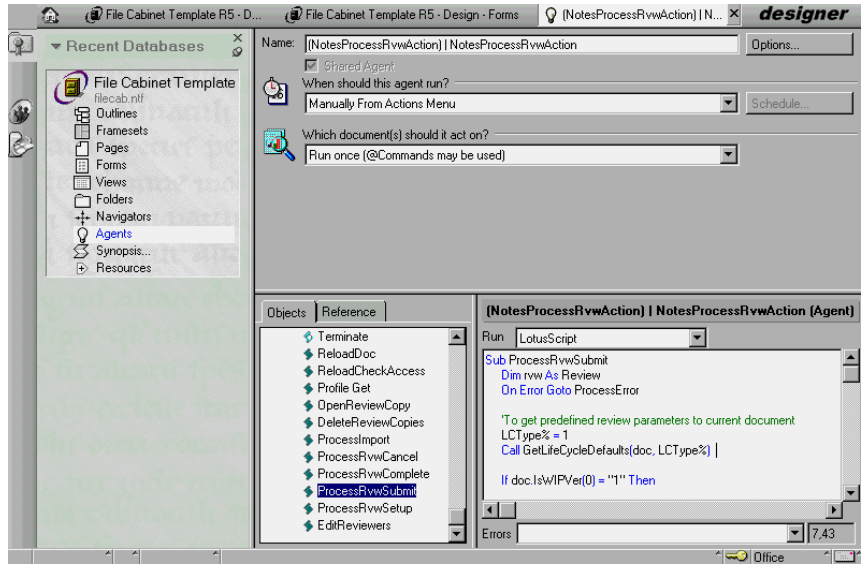


8. From the list, select (NotesProcessRvwAction) and double-click to open it.
9. Select ProcessRvwSubmit from the procedure list. The script for this procedure will be displayed in the programmer's pane.

10. At the top of the existing code, add the following code:

```
LCType% = 1

Call GetLifeCycleDefaults(doc, LCType%)
```



11. Save and close the agent.

By default Domino.Doc presents a dialog box for setting up the review cycle when the user clicks on the hotspot for review. With the changes to the code in the above steps you are bypassing the code that presents the dialog and the default values for review are fetched and used for the actual review cycle.

Approval cycle

Many business documents require approval before they can be finalized. In an approval cycle, one or more people approve or reject a document before anything further happens to it. If the document requires a review cycle, it should be complete before the approval cycle starts.

Any draft editor can submit the document draft for approval provided drafts are not disabled for the document type and an approval cycle is set up for the document type. The document will be routed to different approvers either serially of in parallel, depending on what was set up in the document type. The document status will change to “Pending Approval.” Once the last approver approves the document, it is either checked in as a version or sent back to the initiator, again depending on the setup in the document type. The document status changes to “Approved.” The initiator can change the

approver list or cancel the approval cycle at any time prior to completion if they were given that right in the document type. If an approver rejects a document, the document status will change to “Rejected” and the approval cycle ends.

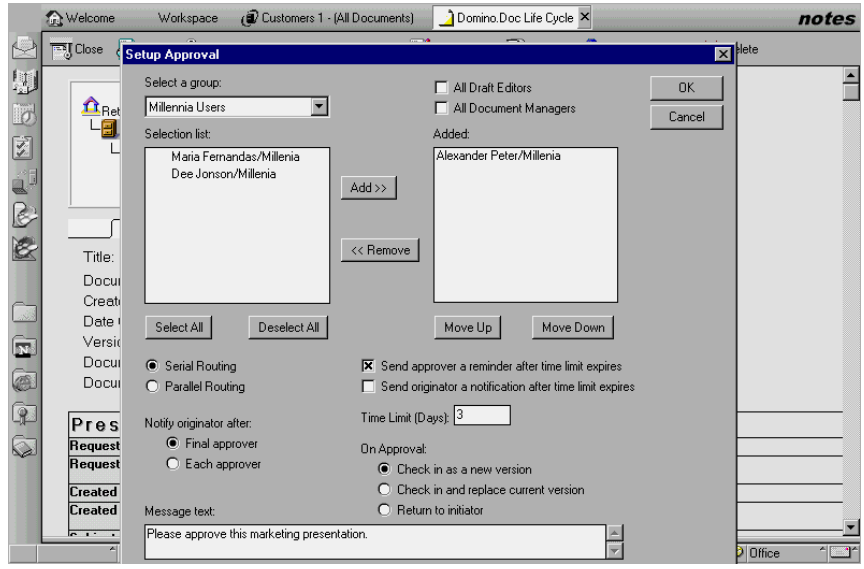
Initiating approval cycle

You can submit only document drafts for approval. If the document you want to submit for approval is currently a version, you should check it out first and then check it back in as a draft.

To initiate the approval cycle, perform the following steps:

1. Open the library by double-clicking the database icon.
2. Open the file cabinet and the binder that contains the document you want to submit for approval.
3. Open the document profile and click Submit for Approval. The Setup Approval dialog box is displayed. The Setup Approval dialog box will have the values specified in the document type as default values. The draft editor can overwrite these values provided they were given the right in the document type.

Note If you are working in a non-master file cabinet, you should first check out the current draft.



4. Specify the document approvers.
   - Select "All draft editors" to make all draft editors approvers.
   - Select "All document managers" to make all document managers approvers.
   - Select "Specify individual approvers" to make some specific individuals approvers and specify their names.

   To specify individual approvers, follow these steps:

   a. Select a group from the list.
   b. The names of group members are displayed in the selection list.
   c. Click Select All if you want all members to be in the approvers' list.
   d. Select individual names and click Add to add each specified name to the approvers' list.
   e. Click Deselect All if you want to clear your selection.
   f. Use the Move Up and Move Down buttons to arrange the members in the order you want them to approve the document. This is important if you select Serial routing type.

   Note It is also possible that users were invited to participate in the file cabinet as individuals, in which case their names will appear in the Selection list automatically, and no groups will need to be selected. However, it is recommended to work with groups in the file cabinets as this normally requires less administrative work.

1. Specify the routing type.
   - Choose "Serial" if you want each approver to approve the document one at a time, in the order you specified in the list.
   - Choose "Parallel" if you want each approver to approve the document at same time.
2. Specify the time limit option.
   - Select "Send approver a reminder after the time limit expires" if you want to notify the approver after a specified time.
   - Select "Send originator a reminder after the time limit expires" if you want to notify the originator after a specified time.
3. Specify whether to notify the originator after each approval or only after final approval.
4. Specify what should happen to the document once the approval cycle is complete: whether to check in the document as a version or to send the document back to the originator.

5. Enter the message to be sent to approvers.
6. Click Save if you are working in a browser.
7. Click OK if you are working in Domino.

The approvers will be notified by e-mail that they have documents to approve. The e-mail will contain a link to the document which the approver can access by clicking on the link.

Predefined approval cycle

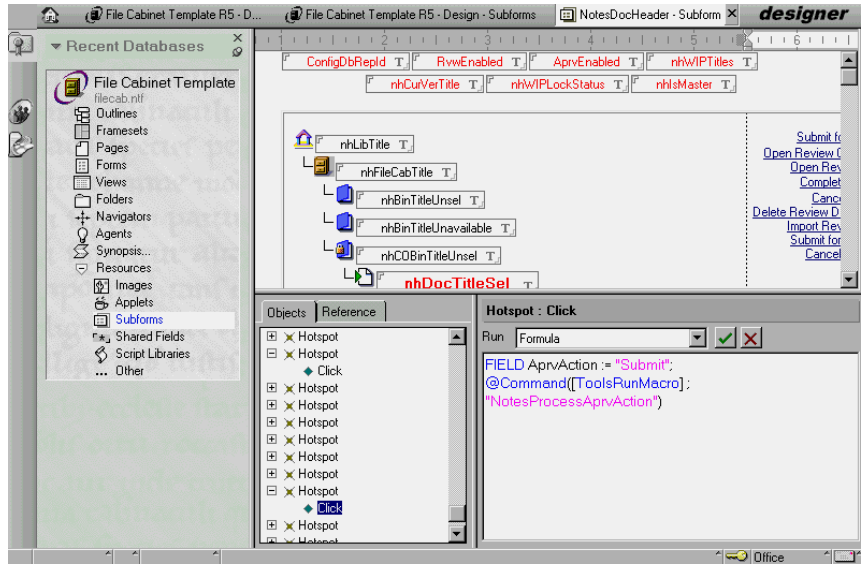
In the previous section, we described an approval cycle in which the draft editors can set up the approval cycle. The draft editors can overwrite the default values specified in the document type. But in some cases, organizations require a predefined approval cycle and all the documents are required to follow the approval cycle defined in the document type. The draft editors should not be able to change this setup.

To hide the Setup Approval dialog box from the draft editors while they are submitting documents for approval, modify the design of the file cabinet template as follows:

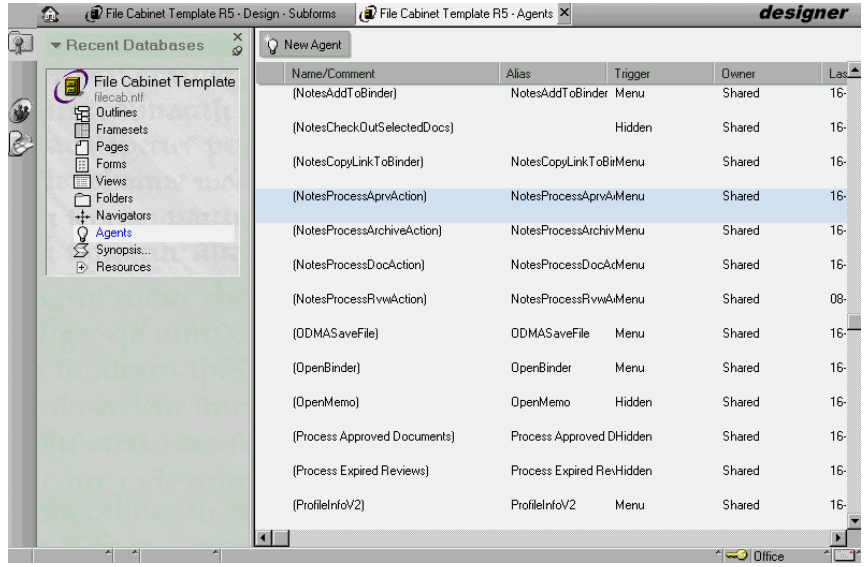
1. Open the File Cabinet Template in Domino Designer.
2. From the design elements list select Resources - Subforms. The list of subforms in the template is displayed.
3. From the list, select NotesDocHeader and double-click to open it.
4. Click the hotspot Submit for Approval. The formula for the hotspot will appear in the formula window.
5. Replace the existing formula with a new formula as follows:

```
FIELD AprvAction := "Submit";

@Command([ToolsRunMacro] ; "NotesProcessAprvAction")
```



6. Save and close the subform.
7. Select Agents from the design elements list. The list of agents in the template is displayed.
8. From the list, select (NotesProcessAprvAction) and double-click to open it.

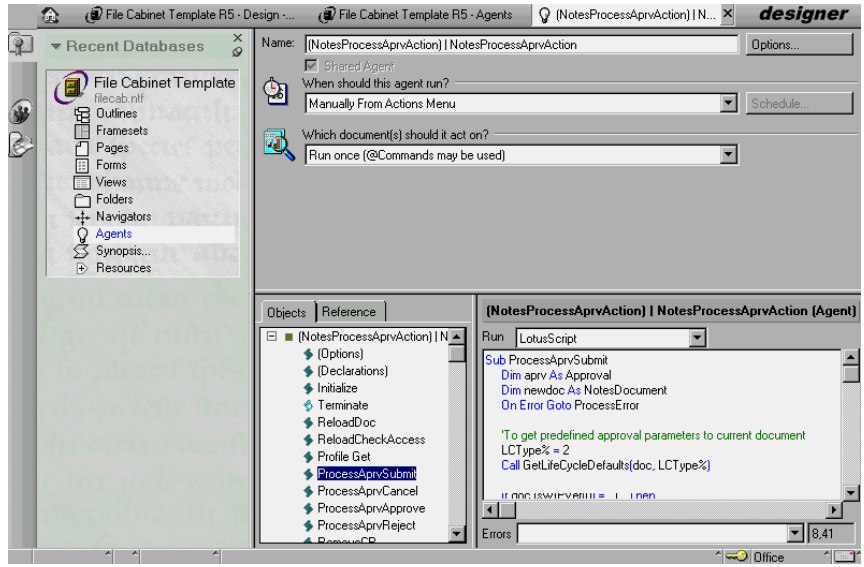


9. Select ProcessAprvSubmit from the procedure list. The script for this procedure will be displayed in the programmer's pane.

10. At the top of the existing code, add the following code:

```
LCType% = 2

Call GetLifeCycleDefaults(doc, LCType%)
```



11. Save and close the agent.

By default Domino.Doc presents a dialog box for setting up the approval cycle when the user clicks on the hotspot for review. With the changes to the code in the above steps you are bypassing the code that presents the dialog and the default values for approval are fetched and used for the actual approval cycle.

Note After modifying the design of a template database, you should refresh the design of databases created with that template for the changes to become effective. To refresh the database design choose File - Database - Refresh Design or type the following command on the server console:

```
load design
```

Review and approval cycles in distributed Domino.Doc setup

If you have a distributed Domino.Doc setup with users connected to master and replica libraries, you have to consider the replication schedule and mail routing interval while setting up the review and approval cycles. If a user working on a replica library sends a document for review or approval to a person connected to the master library and the mail routing takes place before the library replication, the reviewer or approver will get the mail notification with the document link, but they will not be able to access the document until the library replication takes place. So you have to optimize between these two intervals so that reviewers and approvers will get notification only after the library replication

Summary

In this chapter, we discussed the Domino.Doc review and approval cycles. We discussed how to initiate review and approval cycles and how to set a predefined review and approval cycle for documents.

Chapter 9
Domino.Doc events

The Domino.Doc events are used to change the way Domino.Doc reacts at certain key moments. By coding these events you create customized processing to satisfy your business requirements.

This chapter describes how to use events to customize your application.

Customizing events

This section describes what events, DocEvents, and event handlers are, and how you can use them to customize your Domino.Doc applications.

Events

An event is something that "happens" inside Domino programmable objects. You insert code into it and when that certain event occurs your code will run. For example when you click on a button the event "click" occurs and the code associated with this event will run. Events are not limited to Domino.Doc applications. They are available in all Domino applications, but Domino.Doc adds some event handlers that are specialized for document management.

Query and Post events

Query and Post are generally words that precede event names, and they can be interpreted as "Before" and "After" respectively.

For example the event QueryDeleteDocument happens when you try to delete one document in the file cabinet in Domino.Doc, before the document is deleted.

Query events are normally used to determine whether an action really should happen and/or to modify parameters, while Post events are useful for user notifications and the start of subsequent processing.

Event handlers

Event handlers are basically the response that you will get when one specific event occurs.

You can invoke Domino.Doc event handlers for the following events:

- Document creation, check-in/check-out, review and approval (setup, submittal, and completion)
- File cabinet creation, modification, and deletion
- File cabinet contents searches
- File cabinet replication
- File cabinet contents retrieval
- Binder and document type creation, modification, and deletion

Difference between Front-end UI and Back-end objects

In a popular manner, Front-end UI objects deal with things in the Notes client, and the Back-end objects deal with things on the Domino server. Front end UI objects are not supported for the Web client.

For example if a Notes client user opens one document to edit you can use one Front-end object in one event to grab text within one field in the current user's document.

Back-end objects are normally used for Domino element manipulation like a document or a view column, and these objects do not support interaction with the Notes client user interface (screen).

Note Although you can insert code declaring NotesUI objects, do not use them if you have browser access to your Domino.Doc in order to avoid problems.

Domino.Doc events

Domino.Doc provides an extensive event model that enables you to customize the processing that occurs during various Domino.Doc events. Each event has a corresponding event handler: a blank subroutine that you can populate to provide custom processing.

Domino.Doc event handlers

Inside the Domino.Doc templates there are some LotusScript functions, called event handlers, in which you can insert your LotusScript code to customize your application.

In the Library template the event handlers are defined in the LibEvents Script Library, and in the file cabinet template the event handlers are located in the DocEvents Script Library.

The following event handlers are found in the LibEvents Script Library located in the library template; they are also known as "LibEvents":

- SearchStartDisplay
- SearchProcessResults
- SearchEndDisplayFileCabResults
- SearchEndDisplay
- SearchStartDisplayFileCabResults
- QuerySaveDocumentType
- PostSaveDocumentType
- QueryDeleteDocumentType
- QuerySaveBinderType
- PostSaveBinderType
- QueryDeleteBinderType
- PostDeleteBinderType
- QuerySaveFileCabinet
- PostSaveFileCabinet
- QueryDeleteFileCabinet
- PostDeleteFileCabinet
- PostDeleteDocumentType
- QuerySaveReplica
- PostSaveReplica

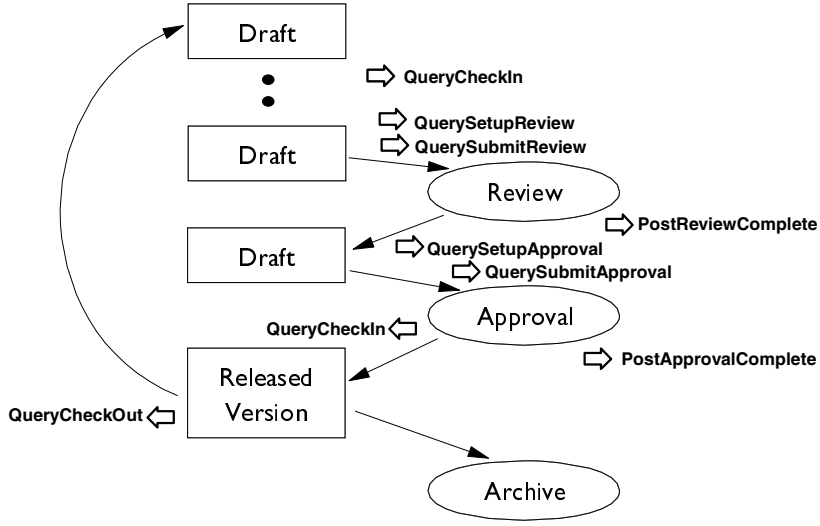
The following event handlers are found in the DocEvents Script Library located in the file cabinet template; they are also known as "DocEvents":

- QueryCheckOut
- PostCheckOut

- QueryCheckIn
- PostCheckIn
- QueryAddToBinder
- PostAddToBinder
- QueryDeleteDocument
- PostDeleteDocument
- QuerySubmitReview
- QuerySubmitApproval
- GetEventErr
- PostReviewComplete
- PostApprovalComplete
- QuerySetupReview
- QuerySetupApproval

Important Do not remove any event handler functions from the Domino.Doc script libraries.

Important The following figure illustrates the relationship between DocEvents occurrences and the document life cycle:



Event handlers and objects

Once you write code in particular events, you are also able to manipulate Domino.Doc-specific objects inside these events. You use the properties and methods of these objects to get and set information about the item being processed.

Domino.Doc provides event handlers for the following objects:

- Documents (DocDocument)
- File cabinets (DocFileCabinet)
- File cabinet contents (SearchDisplayObject)
- File cabinet replication (DocReplica)
- File cabinet retrieval requests (Retrieval)
- Binder types (DocBinderType)
- Document types (DocDocumentType)

See Appendix E: Domino.Doc object model for events, properties, and methods available in Domino.Doc object classes.

Why use LibEvents and DocEvents

There are many different reasons why you would want to use LibEvents and DocEvents, but we are going to focus on DocEvents customization because once you are familiar with Domino.Doc DocEvents custom processing, you will also be able to take advantage of LibEvents for custom processing.

LibEvents are generally used for administrative or document searching purposes.

Some examples of why you would want to use the event handlers for documents are:

- Overriding an event
- Supplementing functions
- Triggering other functions
- Integrating third-party technology

Note There are many other good reasons for using event handlers. Only your imagination sets the limit!

Millennia's DocEvents customization

To satisfy Millennia's requirements, we customized DocEvents to accomplish the following processing steps:

- Publish the presentations
- Notify Web users
- Notify sales staff

Publishing the presentations

Once a presentation document is approved by the final approver, the document contents must be published on the Millennia Web site, in the planets' presentation area.

This requirement is completed by inserting code into the PostApprovalComplete event.

Notifying Web users

According to the preferences that Web users fill in when registering to receive planet information updates, the users will receive weekly e-mail messages about Millennia's Web site updates.

Millennia's Web Site stores users' registration information in a relational database.

We are not going to create a script code in DocEvents to search the user's data in the relational database. This will be done using agents instead, and it is discussed in a later chapter. Now we are only going to use the event handler to mark this document to be processed by the agent.

We will also use the QueryCheckIn event to build this custom application.

Note We cannot use the PostCheckIn event to set the flag value in a field because changes to document fields can only be done to the working copy (the checked out document).

Follow these steps to complete Millennia's requirement to publish approved presentations on their Web site, in the appropriate area, and to flag the Web site's document for user notification:

1. Open the File Cabinet Template in the Domino Designer client.
2. Open the DocEvents Script Library.
3. Find the PostApprovalComplete event.
4. In the entry area of the Programmer's pane insert the following code:

```
' Test if the approved document is a presentation
If (ddoc.DocType Like "Presentation") Then
```

```
            srcfile$ = "c:\temp\" + ddoc.DocFilename

' Extract the source file (e.g. WordPro, Word, etc.)
            Call ddoc.ExtractSourceFile(srcfile$, True)

' Declare variables to perform the search in the site database
            Dim db As New NotesDatabase ("Tyr", "Presenta.nsf")
            Dim col As NotesDocumentCollection
            Dim attname As Variant
            attname=ddoc.GetFieldValue("AttractionName")

' Create a search criteria to search the site
' database for an existing presentation
            srch$ = "Form =" & {"Presentation"} & " & _
AttractionName = "  & {"} & attname(0) & {"}

'Perform a search for existing presentation on the site
' database
            Set col = db.Search( srch$, Nothing, 0)

' Delete any document found that matches the search criteria
            If   Not col  Is Nothing Then
               Call col.RemoveAll( force )
            End If

' Create a new document inside the destination database
            Dim newdoc As New NotesDocument(db)
            Dim rtitem As NotesRichTextItem

' Fill in the document fields
            newdoc.Form="Presentation"
            newdoc.Subject=ddoc.GetFieldValue("Subject")
```

```
            newdoc.Description=ddoc.GetFieldValue("Description")
            newdoc.Locations=ddoc.GetFieldValue("Locations")
            newdoc.Places=ddoc.GetFieldValue("Places")
            newdoc.Attractions=ddoc.GetFieldValue("Attractions")

newdoc.AttractionName=ddoc.GetFieldValue("AttractionName")
            newdoc.Cost=ddoc.GetFieldValue("Cost")
            newdoc.Duration=ddoc.GetFieldValue("Duration")
            newdoc.CreatedOn=Str(Today)

' Attach the file into the document Body field
            Set rtitem = New NotesRichTextItem( newdoc, "Body" )
            Call _
rtitem.EmbedObject(EMBED_ATTACHMENT,"",srcfile$)

' Save the new document
            Call newdoc.Save(True, False)

' Finally, delete the temporary file from the file system
            Kill srcfile$
      End If
```

Notifying sales staff

To keep Millennia's sales representatives informed about new offers and Millennia's Web site updates, the Domino.Doc solution must also notify them whenever new sales presentations are available on the Web.

Use the PostCheckIn event to build this solution.

You will also need to use the QueryCheckOut to set up a flag in a field, because when a document is checked in to the database you cannot guarantee that it will be checked in as a new version unless you specify this.

The best event in which to include this code is the PostApprovalComplete, but for this example we are going to use QueryCheckOut and PostCheckIn events to show you how to use different event handlers to customize your applications.

Follow these steps to create this solution:

1. Open the File Cabinet Template in the Domino Designer client.
2. Open the DocEvents Script Library.
3. Find the QueryCheckIn event.
4. Beneath whatever code is already in the subroutine, insert the following code:

```
' If the document is a presentation set a previous
' version in a field called VersionControl
     If (ddoc.DocType Like "Presentation") Then
          Dim version As Variant
          version = ddoc.GetFieldValue("Version")
          Call _
      ddoc.SetFieldValue("VersionControl", version(0))
     End If
```

5. Insert the following code in the PostCheckin event:

```
' Set the var1 and var2 variables the previous and
' current presentation version respectively
     Dim var1 As Variant
     var1=ddoc.GetFieldValue("VersionControl")

     Dim var2 As Variant
     var2=ddoc.GetFieldValue("Version")

     If (ddoc.DocType Like "Presentation")_
 And (var2(0) > var1(0)) Then
          Dim session As New NotesSession
          Dim db As NotesDatabase
          Set db = session.CurrentDatabase
' Create a new document to send a message to
' The sales people
          Dim doc As notesdocument
```

```
            Set doc = New NotesDocument( db )

' Set form information in order to avoid problems when
' the recipients try to open the mail message
            doc.Form = "Memo"

' Set the recipients list
            doc.SendTo = "Sales"

' Set the message subject
            doc.Subject = _
               "A new presentation is now available"

' Reuse the previous declared variable var1 and var2
' as location and attractions and set values to var3
 ' and var4 as attraction names and travel cost
            var1 = ddoc.GetFieldValue("Locations")
            var2 = ddoc.GetFieldValue("Attractions")
            Dim var3, var4 As Variant
            var3 = ddoc.GetFieldValue("AttractionName")
            var4 = ddoc.GetFieldValue("Cost")

' Set the message body including a doclink to the
' document stored on the Domino.Doc
            Dim msgbody As String
            msgbody = "A new presentation is now available to
the Web users." & Chr$(13) & Chr$(13) &_
"Presentation Summary: " & Chr$(13) &Chr$(9) &_
"Locations: " & var1(0) & Chr$(13) & Chr(9) &_
"Attractions: " & var2(0) & Chr$(13) & Chr$(9) &_
"Name of the attractions: " & var3(0) & Chr(13) & Chr$(9) &_
"Price rate: " & var4(0) & Chr$(13) & Chr$(13) &_
```

```
"Please click on the following link to view this document in
the document in the Millennia's Library Base => "

            Dim rtitem As NotesRichTextItem

            Set rtitem = New NotesRichTextItem( doc, "Body")

            Call rtitem.AppendText(msgbody)

            Call rtitem.AppendDocLink( ddoc.document, "Link to
the Presentation document" )


' Submit the message

            Call doc.Send( False )

      End If
```

Note In this example the recipients, the subject, and the message body were hard coded. However, it is good practice to create one hidden form with access restrictions and one hidden view, to be sure that there will not be more than one document where the code will look up parameters to send messages to the sales representatives. This is good practice because when the parameters are not hard coded you do not need to rewrite your code when things change. In other words, it is easy to maintain the system.

Summary

In this chapter, we presented an overview of the Domino Events. We then identified the event handlers available in Domino.Doc (LibEvents and DocEvents).

We described why you might use the event handlers, and also how to use them in Domino.Doc.

Finally, we described how Millennia's requirements were satisfied using DocEvents.

Chapter 10
Domino.Doc API

The Domino.Doc Application Programming Interface (API) is used to access Domino.Doc libraries when creating external applications, and it is also commonly used within the DocEvents script library. Creating external applications is done to provide customized processing, to create your own user interface, or to bypass the Domino.Doc user interface for background processing of documents (including importing or exporting documents), and bulk-handling of document collections.

This chapter explores three main topics:

- Why use the Domino.Doc API
- Object Model Overview
- Developing Applications with the Domino.Doc API

This chapter does not go into detail about each of the model's classes; however, Appendix F: Domino.Doc API objects and classes contains listings of all of the classes and their respective properties and classes.

By the end of this chapter, a complete custom application will be built using the Domino.Doc API and Microsoft Visual Basic.

Why use the Domino.Doc API

The Domino.Doc API is useful in creating custom applications. Some instances where you would use the Domino.Doc API include:

- Creating a gateway between Domino.Doc and another application
- Creating a custom application that requires direct access to a Domino.Doc library
- Creating a custom application that processes documents in a Domino.Doc library, such as automatically modifying document fields
- Creating a custom application to generate reports based on a Domino.Doc library

In our example scenario of Millennia Space Travel, a Visual Basic application will be created that imports documents into the Millennia library using a customized client.

What you need to use the Domino.Doc API

In order to develop and use applications that utilize the Domino.Doc API, you have to have installed the API (and optionally the Desktop Enabler) on your system using the installation tool.

The Domino.Doc API uses OLE automation enabled code that resides in Windows DLL files, hence it is available for use only within a 32-bit Microsoft Windows environment (such as Windows 9x or Windows NT 4.0 and higher) and be accessed from development tools that support OLE Automation (such as LotusScript, Microsoft Visual Basic, and Microsoft Visual C++).

Installing the Domino.Doc API

The first step in creating a Domino.Doc API application is to install the Domino.Doc API itself. If you have installed the Domino.Doc Desktop Enabler, the Domino.Doc API is already installed, and you can skip to the next section.

To install the Domino.Doc API, use the following steps:

1. From the Domino.Doc library, choose Library Administration.
2. Choose Download Client Software.
3. Launch the attached file (ddsetup.exe).
4. Choose custom install.
5. Choose to install only the Domino.Doc API.
6. Allow the setup program to complete and restart the computer if you are prompted to do so.

Towards the end of this chapter we describe all the steps required to set up a Visual Basic project for creating a stand-alone application that uses the Domino.Doc API.

Object model overview

This section outlines the Domino.Doc API object model. In order to understand each object class, you must first understand the hierarchy of the model.

The object classes relate to each other in the following manner:

- The Library object contains the Rooms and Cabinets collections, as well as FavoriteDocuments and CheckedOutDocuments collections.
- Each Room object contains a Cabinets collection.

- Each Cabinet object contains a collection of Binders, Rooms collections, as well as DocumentProfiles and BinderProfiles collections.
- Each Binder object contains a collection of Documents, as well as a Profile and Security object.
- Each Document object contains a Profile and Security object.
- Each Security object contains a collection of Users.
- Each Profile object contains a collection of Fields.
- Each Field object *may* contain a collection of Keywords (if the field type is keyword).

Library class

In order to access any of the Domino.Doc API classes, you must begin by creating the API object, and then proceed creating child objects (based on the object model hierarchy). After creation of the API object you create the Library object to access information in Domino.Doc.

Once you have created the Domino.Doc API object, getting the library is a two-step process: first you must use the Domino.Doc API object to set the login ID and password to be used, and second, you must create the Library Object.

Note The authentication process does not actually occur until a library property or method is accessed.

Important Be sure to use a login ID that has permission to access the parts of the library that you plan to access with your application. A simple test is to manually perform the steps that your application will perform, and if your ID does not have the required access to do it manually, you will not be able to do it programmatically either.

There are two ways to access the library using the Domino.Doc API:

- Using the Hyper Text Transfer Protocol (HTTP)
- Using the Notes Protocol

Important If you are using HTTP to access the library, be sure that the ID you are using has an Internet password set, otherwise authentication will fail. To test this, try accessing the library from a Web client. If you cannot access the library from a Web client, your program will not have the necessary access either.

In the Millennia Space Travel example, HTTP will be used instead of the Notes protocol. The only difference, from an API coding perspective, is the one line that sets the login and password information.

The following line sets the login information for the hyper text transfer protocol:

```
Call theApi.SetHttpLogin ("Siâms Lewys", "red")
```

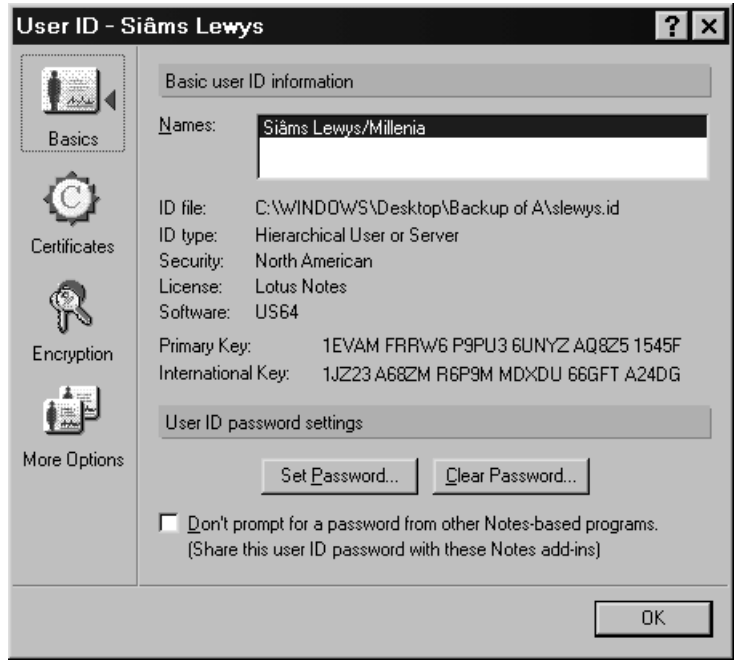
where the first parameter is the user name, and the second is the Web password.

The following line sets the login information for the Notes protocol:

```
Call theApi.SetNotesLogin ("red")
```

where the parameter is the Notes password of the current Notes ID.

Note When using the Notes Protocol, you can omit the login step if the current Notes ID is set to share its password with Notes add-ins. You can modify your Notes client so that the password is shared by selecting the File menu, then Tools, and then User ID. When the User ID box appears, select "Don't prompt for a password from other Notes-based programs" (as shown in the following figure).



The following examples demonstrate creating a Library object using LotusScript, Visual Basic, and Visual C++.

Important No matter which development tool you use, you will need to tell that tool to use the Domino.Doc API. How this is done depends on the individual tool used. The examples in this section outline how to access the Domino.Doc API using LotusScript, Visual Basic, or Visual C++.

Accessing the Library with LotusScript

To create the Library with LotusScript, complete the following steps:

1. Create a new agent in a database (it doesn't need to be a Domino.Doc database, as the Domino.Doc API will be accessing the library by specifying its URL).
2. Insert the following code, changing the login ID, password, and library URL to match your system:

```
'Declare all API objects as Variants


Dim theApi As Variant
Dim theLibrary As Variant


'Create the API object
Set theApi = CreateObject("DominoDoc.API")


'Set the login name and password (change these to your
'name and your password!)
Call theApi.SetHttpLogin("Siâms Lewys", "red")


'Get the Library object from the API (change this to
'the URL of your Library!)
Set theLibrary = theApi.GetLibrary("http://tyr.lotus.com
                 /domdoc/milleniaLib.nsf")
```

Accessing the Library with Visual Basic

To create the Library object with Visual Basic, complete the following steps:

1. Create a new project.
2. From the Project menu, select References. When the References window appears, add **`Lotus Domino.Doc`**.
3. Create a button on a form.
4. Double-click the button to access the button's code.
5. Insert the following code:

```
Private Sub Command1_Click()
```

```
'Declare the API objects
Dim theApi As Api
Dim theLibrary As Library

'Create the API object
Set theApi = New Api

'Set the login name and password (change these to your
'name and your password!)
Call theApi.SetHttpLogin ("Siâms Lewys", "red")

'Get the Library object from the API (change this to
'the URL of your Library!)
Set theLibrary = theApi.GetLibrary("http://tyr.lotus.com/
                 domdoc/milleniaLib.nsf")

End Sub
```

Accessing the Library with Visual C++

To create the Library object with Visual C++, complete the following steps:

1. Create a new project.
2. Choose MFC AppWizard (exe), and call it **`Test`**.
3. In step 3 of 6, in the section "What other support would you like to include?", check Automation.
4. Replace **`CTestApp::OnAppAbout`** with the code (shown below these steps).
5. Customize the login name, password, and the library URL for your system.
6. Run the class wizard.
7. Select Add Class from the Automation page.
8. Select From a type library…
9. In the Import from Type Library dialog, find the directory that has the file ddoc2api.tlb, and select that file.

10. In the Confirm Classes dialog, highlight all of the classes, and click OK.
11. Insert #include**ddoc2api.h** under #include **test.h**.

```
void CTestApp::OnAppAbout()
{
      // Declare the API objects
      IApi theApi;
      ILibrary theLibrary;

      // Create the API object
      if (!theApi.CreateDispatch("DominoDoc.Api"))
      {
            AfxMessageBox("Error creating Domino.Doc API
                        object!");
            return;
      }

      // Set the login name and password (change these to your
      // name and your password!)
      theApi.SetHttpLogin("Siâms Lewys", "red");

      // Get the Library object from the API (change this to
      // the URL of your Library!)
      theLibrary.AttachDispatch(theApi.GetLibrary("http://
                  tyr.lotus.com/domdoc/milleniaLib.nsf"));
}
```

Collection classes

The objects that you retrieve through the Domino.Doc API are contained in collections. The collection classes in the Domino.Doc API are as follows:

- Rooms
- Cabinets
- Binders
- Documents
- Profiles

- Fields
- Keywords
- Users
- Formats

Common collection class properties and methods

Each collection class has properties and methods that allow you to list the items in the collection, to retrieve a particular item from the collection, or to manipulate the collection.

Ways to access an object in a collection

There are four ways to retrieve an object from a collection. These are:

- Position
- Property
- Variant
- Default

Access by position

You can access an object by its position by using the ItemByIndex property and you can use the Count property.

The ItemByIndex property is used to retrieve an item by its position within a collection. For example, the ItemByIndex property could be used to get the fourth binder from the second file cabinet.

Note Access by position is always zero-based. In other words, the first document would be document number zero, and the last document would be a number equal to the total number of documents minus one.

The Count property can be used with the ItemByIndex property to iterate through the objects in a collection. For example, the two could be combined to list each document title in a binder.

Access by item property

An object within a collection class can be accessed based on a certain attribute that the object has. This is done using the ItemByTitle property, ItemByName property, or ItemByValue property, depending on which Class is being used.

Access by variant

The Item property allows you to access a collection object by variant. This works by checking to see whether the variant contains a numeric value or a string value. If it contains a numeric value, then it is assumed to represent the index of the item to retrieve. If it contains a string, then it is assumed to represent either the Title property or the Name property (depending on the class).

Access by default property
All collection classes have the Item property as the default property. What this means is that whenever you use the Item property, the word "Item" may be omitted.

In other words, the following line:

```
Set theCabinet = theLibrary.Cabinets.Item("Presentations")
```

is the same as the following line:

```
Set theCabinet = theLibrary.Cabinets("Presentations")
```

Rooms collections and Room objects

The Domino.Doc API has a Rooms collection class and a Room object class, which correspond to the file rooms within the Domino.Doc library.

The following are points to remember when using the Room and Rooms classes:

- You can obtain a list of rooms that a file cabinet is associated with using the Rooms property of any Cabinet object.
- You can obtain a list of file cabinets that a room is associated with using the Cabinets property of any Room object.
- You can obtain a list of rooms in a library.
- File cabinets may belong to more than one file room.
- You cannot create or remove a file room through the Domino.Doc API.
- You can obtain a list of cabinets in a library (hence, the use of file rooms through the Domino.Doc API is optional).

Accessing the Rooms collection class with Visual Basic

This example demonstrates the following:

- It gets the Library object from the Domino.Doc API.
- It gets the Rooms collection from the Library object.
- It gets each Room object in the Rooms collection using the ItemByValue property.
- It gets the name of each file room, and adds it to a Visual Basic Combo Box called cmbFileRooms.

To access the Rooms collection class with Visual Basic, use the following code:

```
Private Sub Command1_Click()

      'Declare the API objects
      Dim theApi As Api
      Dim theLibrary As Library
      Dim theRooms As Rooms
      Dim theRoom As Room

      'Create the API object
      Set theApi = New Api

      'Set the login name and password (change these to your
      'name and your password!)
      Call theApi.SetHttpLogin ("Siâms Lewys", "red")

      'Get the Library object from the API (change this to
      'the URL of your Library!)
      Set theLibrary = theApi.GetLibrary("http://tyr.lotus.com/
                       domdoc/milleniaLib.nsf")

      'Get the Rooms object from the Library
      Set theRooms = theLibrary.Rooms

      'Go through each Room in the Rooms
      '(remember that the Count property gives the
      ' number of file cabinets, but the ItemByIndex
      ' has the first item as number zero, so
      ' the last item would be the count minus one)
      iLastRoom = theRooms.Count - 1
      For i = 0 To iLastRoom
            'Get the Room number 'i' from the Rooms
```

```
            Set theRoom = theRooms.ItemByIndex(i)
            'Add the title of the room to the Combo Box
            'called cmbRooms (be sure you have a Combo
            'box created with this name!)
            cmbRooms.AddItem theRoom.Title
        'Get the next Room
        Next i


    End Sub
```

Cabinets collections and Cabinet objects

The Domino.Doc API has a Cabinets collection class and a Cabinet object class, which correspond to the file cabinets within the Domino.Doc library.

The following are points to remember when using the Cabinet and Cabinets classes:

- You can obtain a list of cabinets in a library.
- A Cabinet contains a Binders collection.
- A Cabinet contains all of the available profiles for both binders and documents.
- File cabinets may belong to more than one file room.
- You cannot create or remove a file cabinet through the Domino.Doc API.

Accessing the Cabinets collection class with Visual Basic

This example demonstrates the following:

- It gets the Library object from the Domino.Doc API.
- It gets the Cabinets collection from the Library object.
- It gets each Cabinet object in the Cabinets collection using the ItemByValue property.
- It gets the name of each file cabinet, and adds it to a Visual Basic Combo Box called cmbFileCabs.

To access the Cabinets collection class with Visual Basic, use the following code:

```
Private Sub Command1_Click()

      'Declare the API objects
      Dim theApi As Api
      Dim theLibrary As Library
      Dim theCabinets As Cabinets
      Dim theCabinet As Cabinet

      'Create the API object
      Set theApi = New Api

      'Set the login name and password (change these to your
      'name and your password!)
      Call theApi.SetHttpLogin ("Siâms Lewys", "red")

      'Get the Library object from the API (change this to
      'the URL of your Library!)
      Set theLibrary = theApi.GetLibrary("http://tyr.lotus.com/
                       domdoc/milleniaLib.nsf")

      'Get the Cabinets object from the Library
      Set theCabinets = theLibrary.Cabinets

      'Go through each Cabinet in the Cabinets
      '(remember that the Count property gives the
      ' number of file cabinets, but the ItemByIndex
      ' has the first item as number zero, so
      ' the last item would be the count minus one)
      iLastFileCab = theCabinets.Count - 1
      For i = 0 To iLastFileCab
```

```
            'Get the Cabinet number 'i' from the Cabinets
            Set theCabinet = theCabinets.ItemByIndex(i)
            'Add the title of the cabinet to the Combo Box
            'called cmbFileCabs (be sure you have a Combo
            'box created with this name!)
            cmbFileCabs.AddItem theCabinet.Title
        'Get the next Cabinet
        Next i


End Sub
```

Binders collections and Binder objects

This section outlines how to use a Binders collection and a Binder object through the Domino.Doc API.

> **Note** You can use the Domino.Doc API to create and delete binders, but not to modify existing ones.

Creating and deleting Binders

To create a binder using the Domino.Doc API, use the following steps:

1. Create a new Binder object using the Add method of the Binders collection.
2. Set the name of the binder using the Title property of the Binder object.
3. Commit this to the library using the Save method of the Binder object.
4. Check the binder into the library using the CheckIn method of the Binder object (which uses two parameters, *action&* which is always 2 for check in as new, and *comment$* which is any string). See the example below for details.

   > **Important** Make sure the ID that you use has permission to create binders in the file cabinet.

To delete a binder using the Domino.Doc API, the Delete method of the Binder object is used.

> **Note** You cannot delete a binder if it is checked out or if it contains documents. If the binder is checked out, it must be checked in prior to deleting. If there are documents within the binder, they must be deleted prior to the binder being deleted.

> **Important** Make sure the ID that you use has permission to delete binders in the file cabinet.

The following example shows the creation of a binder and the deletion of another using Visual Basic:

```
Private Sub Command1_Click()

      'Declare the API objects
      Dim theApi As Api
      Dim theLibrary As Library
      Dim theCabinets As Cabinets
      Dim theCabinet As Cabinet
      Dim theBinders As Binders
      Dim theNewBinder As Binder
      Dim theOldBinder As Binder

      'Create the API object
      Set theApi = New Api

      'Set the login name and password (change these to your
      'name and your password!)
      Call theApi.SetHttpLogin ("Siâms Lewys", "red")

      'Get the Library object from the API (change this to
      'the URL of your Library!)
      Set theLibrary = theApi.GetLibrary("http://tyr.lotus.com/
                       domdoc/milleniaLib.nsf")

      'Get the Cabinets collection from the Library
      Set theCabinets = theLibrary.Cabinets

      'Get the Cabinet where the binder will be created
      Set theCabinet = theCabinets.ItemByTitle("Presentations")
```

```
    'Get the Binders collection from the Cabinet
    Set theBinders = theCabinet.Binders

    'Create a new Binder object in the Binders collection
    Set theNewBinder = theBinders.Add

    'Set the Title property of the new Binder object
    theNewBinder.Title = "Mars"

    'Save the new Binder object
    Call theNewBinder.Save

    'Check in the new binder
    'Note the syntax and parameters:
    'Call binder.CheckIn(action&, comment$)
    'where binder is the Binder object
    '      action& is always 2 (for check in as new)
    '      comment$ is any text string
    Call theNewBinder.CheckIn(2, "Redbook API example")

    'Get the Binder object to be deleted
    Set theOldBinder = theBinders.ItemByTitle("Earth")

    'Delete the Binder object
    Call theOldBinder.Delete

End Sub
```

Documents collections and Document objects

The Domino.Doc API uses the Documents collection and the Document object to represent documents in the system.

The four main procedures performed with the Document object are:

- Creating documents
- Getting documents
- Updating documents
- Deleting documents

Creating documents

The Domino.Doc API is used to programmatically create new Domino.Doc documents.

There are five steps that are generally followed to create a new document. These steps are:

1. Use the Add method of the Documents collection to create a new empty document.
2. Use the SetContents method of the new Document object to attach a file.
3. Set the Title property of the Document object to give it the name to be used by Domino.Doc.
4. Use the Save method of the Document object to place the new document in the library.
5. Use the CheckIn method of the Document to check in the document.

   Note The new document will be saved in the parent binder of the documents collection, unless that binder is checked out or the user does not have the appropriate rights. If the parent binder cannot be used to store the new document, then the file cabinet's default binder is used.

The following example shows the creation of a document using Visual Basic:

```
Private Sub Command1_Click()


    'Declare the API objects
    Dim theApi as Api
    Dim theLibrary as Library
    Dim theCabinets as Cabinets
    Dim theCabinet as Cabinet
    Dim theBinders as Binders
```

```
Dim theBinder as Binder
Dim theDocuments as Documents
Dim theDocument as Document

'Create the API object
Set theApi = New Api

'Set the login name and password (change these to your
'name and your password!)
Call theApi.SetHttpLogin ("Siâms Lewys", "red")

'Get the Library object from the API (change this to
'the URL of your Library!)
Set theLibrary = theApi.GetLibrary("http://tyr.lotus.com/
                 domdoc/milleniaLib.nsf")

'Get the Cabinets collection from the Library
Set theCabinets = theLibrary.Cabinets

'Get the Cabinet from the Cabinets collection
Set theCabinet = theCabinets.ItemByTitle("Presentations")

'Get the Binders collection from the Cabinet
Set theBinders = theCabinet.Binders

'Get the Binder where the Document is to be created
Set theBinder = theBinders.ItemByTitle("Uranus")

'Get the Documents collection from the Binder
Set theDocuments = theBinder.Documents

'Create the new Document object
```

```
        Set theDocument = theDocuments.Add

        'Add an attachment where the path of the file to be
        'attached has been entered into a field called txtPath
        Call theDocument.SetContents(txtPath)

        'Set the title of the document based on text entered
        'into a field called txtName
        theDocument.Title = txtName

        'Save the Document object into the Domino.Doc library
        Call theDocument.Save

        'Check in the document using the CheckIn method
        'the syntax is as follows:
        'Call document.CheckIn(revisiontype&, action&,
        '     deletedrafts, comment$)
        'where document is a defined Document object
        '      revisiontype& is 1=version or 2=draft
        '      action& is 0=discard, 1=replace, or 2=new revision
        '      deletedrafts is True=delete the drafts, or
        '           False=keep drafts (Note 1 or 0 in LotusScript)
        '      comment$ is a comment about this revision
        'In this example, it is checked in as a new draft:
        Call theDocument.CheckIn(2, 2, False, "Created by Red
                        Book example")

End Sub
```

Getting documents

A document from the library is obtained from a Documents collection. The Documents collection may be obtained from a variety of sources (in other words, a Document object may be created from a number of different types of parent objects). Depending on how the Document object was obtained, only certain methods and properties are valid.

The different types of Document objects are as follows:

- Normal Document or Bookmark (the Documents collection is from a Binder)
- Direct Access
- Search Results
- Favorites
- Checked Out Links
- Resolved Links
- Working Copy
- Review Copy

Normal document or bookmark

A normal document or bookmark is a Document object created in a Documents collection that was created from a Binder object (a bookmark is simply a link to a normal document in another binder). This type of Document object is created by using a line similar to the following:

```
set theDocument = theBinder.Documents.ItemByTitle(title$)
```

where *title$* is the name of the desired document.

The following example demonstrates accessing a document by creating a normal Document object using Visual Basic:

```
Private Sub Command1_Click()


    'Declare the API objects
    Dim theApi as Api
    Dim theLibrary as Library
    Dim theCabinets as Cabinets
    Dim theCabinet as Cabinet
    Dim theBinders as Binders
    Dim theBinder as Binder
```

```
Dim theDocuments as Documents
Dim theDocument as Document

'Create the API object
Set theApi = New Api

'Set the login name and password (change these to your
'name and your password!)
Call theApi.SetHttpLogin ("Siâms Lewys", "red")

'Get the Library object from the API (change this to
'the URL of your Library!)
Set theLibrary = theApi.GetLibrary("http://tyr.lotus.com/
                 domdoc/milleniaLib.nsf")

'Get the Cabinets collection from the Library
Set theCabinets = theLibrary.Cabinets

'Get the Cabinet from the Cabinets collection
Set theCabinet = theCabinets.ItemByTitle("Presentations")

'Get the Binders collection from the Cabinet
Set theBinders = theCabinet.Binders

'Get the Binder where the document exists
Set theBinder = theBinders.ItemByTitle("Uranus")

'Get the Documents collection from the Binder
Set theDocuments = theBinder.Documents
```

```
    'Get a specific document from the collection
    Set theDocument = theDocuments("Great Deals on Uranus")

End Sub
```

Direct Access

The GetDocumentById method of accessing a Library object is used to get a single document by using the unique identifier of the document. This type of Document object is created by using a line similar to the following:

```
set theDocument = theLibrary.GetDocumentById(docid$)
```

where *docid$* is a string containing the document ID of the desired document.

Search results

The SearchDocumentByID method of accessing either a Library object or a Cabinet object returns a Documents collection. This type of Document object is created by using code similar to the following:

```
set theDocuments = theLibrary.SearchDocuments(searchString$,
limit&, sortOrder&, options&)

set theDocument = theDocuments.ItemByTitle(title$)
```

where the parameters are as follows:

- *searchString$* is the string used for the full text query.
- *limit&* is the number of documents to be returned (or 0 to receive all matching documents).
- *sortOrder&* is 8 (or FT_SCORES in LotusScript) to sort by relevance, 32 (or FT_DATE_DES in LotusScript) to sort by decscending date, or 64 (or FT_DATE_ASC in LotusScript) to sort by ascending date.
- *options&* is 512 (or FT_STEMS in LotusScript) to use stem words as the basis of the search, or 1024 (or FT_THESAURUS in LotusScript) to use the thesaurus to search.

The following example demonstrates using search results to get a Documents collection using Visual Basic:

```
Private Sub Command1_Click()

    'Declare the API objects
    Dim theApi as Api
    Dim theLibrary as Library
```

```
Dim theDocuments as Documents
Dim strSearchString As String

'Create the API object
Set theApi = New Api

'Set the login name and password (change these to your
'name and your password!)
Call theApi.SetHttpLogin ("Siâms Lewys", "red")

'Get the Library object from the API (change this to
'the URL of your Library!)
Set theLibrary = theApi.GetLibrary("http://tyr.lotus.com/
                 domdoc/milleniaLib.nsf")

'Search for all documents in the Library, except for
'working copies, old versions, binders, links, and
'documents with only Notes content

strAppendToSearchString = "and(NOT([DocFormat]CONTAINS"
                 + Chr(34) + "NOTES" + Chr(34) + "))"

Set theDocuments = theLibrary.SearchDocuments("*" +
                 strAppendToSearchString, 0, 0, 0)

End Sub
```

Favorites

The FavoriteDocuments property of the Library object returns a collection of favorite documents for the current user. The document objects are links to the actual document.

The following line demonstrates getting a document from the FavoriteDocuments collection:

```
set theDocument =
          theLibrary.FavoriteDocuments.ItemByTitle(title$)
```

where *title$* is the name of the desired document.

Checked-out links

The CheckedOutDocuments property of the Library object returns the collection of documents that the current user has checked out. The following line demonstrates how this is done:

```
set theDocument =
          theLibrary.CheckedOutDocuments.ItemByTitle(title$)
```

Resolved links

To access a document represented by a link (bookmarks, favorites, and checked out links) from the Domino.Doc API, the ResolveLink method of accessing the Document object is used.

Important The Document object that the ResolveLink method is run on must be a link. The Document object that is returned is the actual document represented by the link.

The following line demonstrates how to access a document represented by a link:

```
set theDocument = theLinkDocument.ResolveLink
```

where theLinkDocument is the link, and theDocument is the document represented by the link.

Working copy

The SetToWorkingCopy method of accessing a Document object changes the current object to refer to the working copy. It automatically checks out the document to the current user if it is not already checked out.

The following line demonstrates how to access the working copy of a document:

```
call theDocument.SetToWorkingCopy
```

where theDocument is a defined Document object.

Review copy

The SetToReviewCopy method of accessing a Document object changes the current object to refer to the review copy. This is valid only for a document in review with a review copy for the current user. The user must be a current reviewer.

The following line demonstrates how to access the review copy of a document:

```
call theDocument.SetToReviewCopy
```

Updating documents

The Domino.Doc API is used to modify a document's title, profile, security, or the document's contents (file attachment).

Note In order to make changes to a document, you must have it checked out to you, and it must be set to be the working copy.

The following example demonstrates modifying the document's contents using Visual Basic:

```
Private Sub Command1_Click()

    'Declare the API objects
    Dim theApi as Api
    Dim theLibrary as Library
    Dim theCabinets as Cabinets
    Dim theCabinet as Cabinet
    Dim theBinders as Binders
    Dim theBinder as Binder
    Dim theDocuments as Documents
    Dim theDocument as Document

    'Create the API object
    Set theApi = New Api

    'Set the login name and password (change these to your
    'name and your password!)
    Call theApi.SetHttpLogin ("Siâms Lewys", "red")

    'Get the Library object from the API (change this to
    'the URL of your Library!)
```

```
Set theLibrary = theApi.GetLibrary("http://tyr.lotus.com/
                 domdoc/milleniaLib.nsf")

'Get the Cabinets collection from the Library
Set theCabinets = theLibrary.Cabinets

'Get the Cabinet from the Cabinets collection
Set theCabinet = theCabinets.ItemByTitle("Presentations")

'Get the Binders collection from the Cabinet
Set theBinders = theCabinet.Binders

'Get the Binder where the Document is to be created
Set theBinder = theBinders.ItemByTitle("Neptune")

'Get the Documents collection from the Binder
Set theDocuments = theBinder.Documents

'Create the new Document object
Set theDocument = theDocuments("Cruise Neptune
                  on New Years")

'This sets theDocument to the working copy
'It checks it out first if it is not checked out
Call theDocument.SetToWorkingCopy

'Set the pathName to use the current attachment name
pathName = "C:\" & theDocument.filename

'Copy the attachment from the library to pathName
Call theDocument.GetContents(pathName)
```

```
        'Give a message box telling users that they can modify
        'the document, and to click OK when done
        MsgBox ("You may now make changes to: " & pathName
                        & " Click OK when complete.")

        'Get the modified file and attach re-attach it to the
        'Document object
        Call theDocument.SetContents(pathName)

        'Save the document
        Call theDocument.Save

        'Check in the document using the CheckIn method
        'the syntax is as follows:
        'Call document.CheckIn(revisiontype&, action&,
        '     deletedrafts, comment$)
        'where document is a defined Document object
        '      revisiontype& is 1=version or 2=draft
        '      action& is 0=discard, 1=replace, or 2=new revision
        '      deletedrafts is True=delete the drafts, or
        '           False=keep drafts (Note 1 or 0 in LotusScript)
        '      comment$ is a comment about this revision
        'In this example, it is checked in as a new draft:
        Call theDocument.CheckIn(2, 2, False, "Created by Red
                        Book example")

End Sub
```

Deleting documents

The Domino.Doc API is used to delete documents and bookmarks. Some things to consider when deleting documents and bookmarks using the Domino.Doc API are the following:

- Deleting a bookmark removes only the bookmark, not the document it references.
- Deleting a document removes a document object and all of its associated versions and drafts.
- If the document is referenced somewhere else (such as in a favorites list or a checked out list), deleting the document also removes these references.
- Only managers of a document have the access required to delete a document or a bookmark.
- A document object that is set to Review Copy cannot be deleted.

To programmatically delete a document, use the DeleteAllRevisions method of accessing the Document object. This can only be invoked on documents or bookmarks.

The following example illustrates deleting a document from a binder.

```
Private Sub Command1_Click()


    'Declare the API objects
    Dim theApi as Api
    Dim theLibrary as Library
    Dim theCabinets as Cabinets
    Dim theCabinet as Cabinet
    Dim theBinders as Binders
    Dim theBinder as Binder
    Dim theDocuments as Documents
    Dim theDocument as Document


    'Create the API object
    Set theApi = New Api
```

```
        'Set the login name and password (change these to your
        'name and your password!)
        Call theApi.SetHttpLogin ("Siâms Lewys", "red")

        'Get the Library object from the API (change this to
        'the URL of your Library!)
        Set theLibrary = theApi.GetLibrary("http://tyr.lotus.com/
                         domdoc/milleniaLib.nsf")

        'Get the Cabinets collection from the Library
        Set theCabinets = theLibrary.Cabinets

        'Get the Cabinet from the Cabinets collection
        Set theCabinet = theCabinets.ItemByTitle("Presentations")

        'Get the Binders collection from the Cabinet
        Set theBinders = theCabinet.Binders

        'Get the Binder where the Document is to be created
        Set theBinder = theBinders.ItemByTitle("Uranus")

        'Get the Documents collection from the Binder
        Set theDocuments = theBinder.Documents

        'Create the new Document object
        Set theDocument = theDocuments("Utopian Uranus")

        'Delete the document
        theDocument.DeleteAllRevisions

    End Sub
```

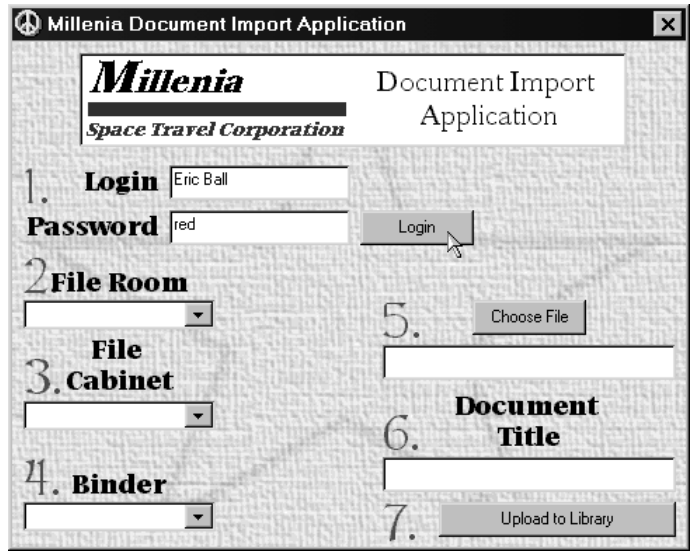
Creating an external application with the Domino.Doc API

This section outlines how the Domino.Doc API is used to create a custom application for Millennia Space Travel Corporation. The application is written in Microsoft Visual Basic, using the Domino.Doc API.

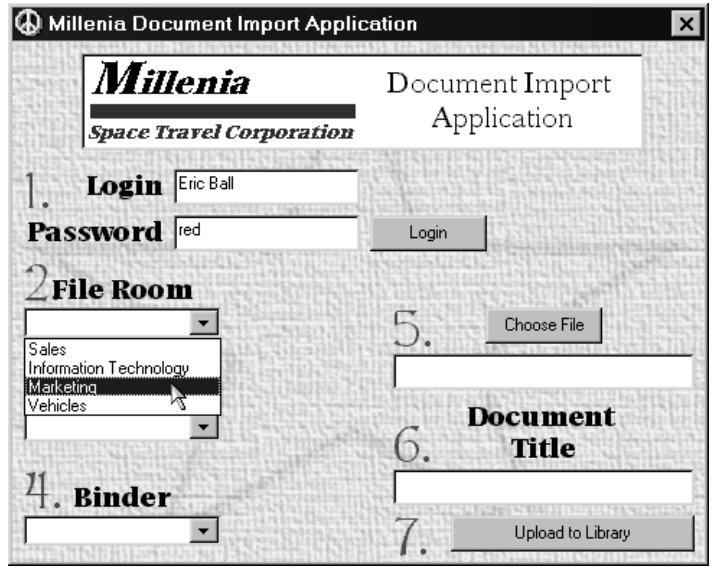
Overview of the application

The custom application will be used to upload a file to the Domino.Doc library.

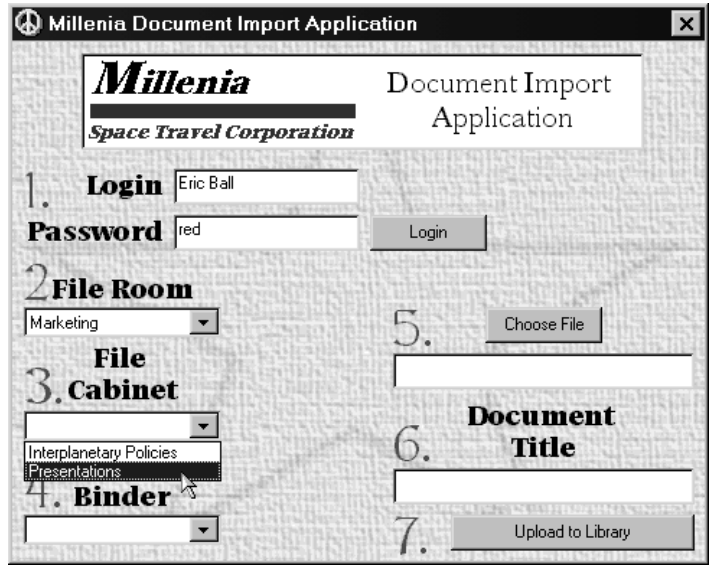
Upon starting the application, the user will enter his or her user name and password, and click on a button labeled "Login." The following figure shows how this is to look.



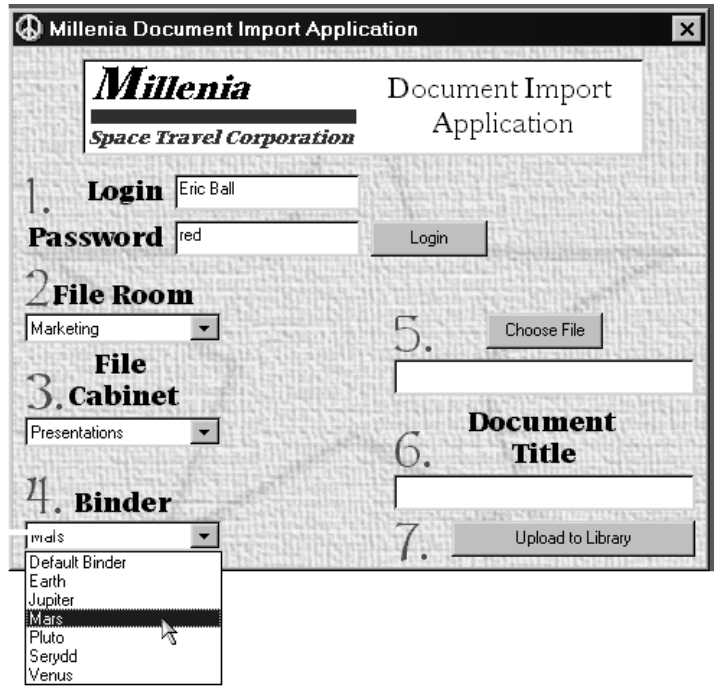
After the user successfully logs in, a combo box will be populated with the names of the file rooms in the library. The user will then select a file room, as shown in the following figure.



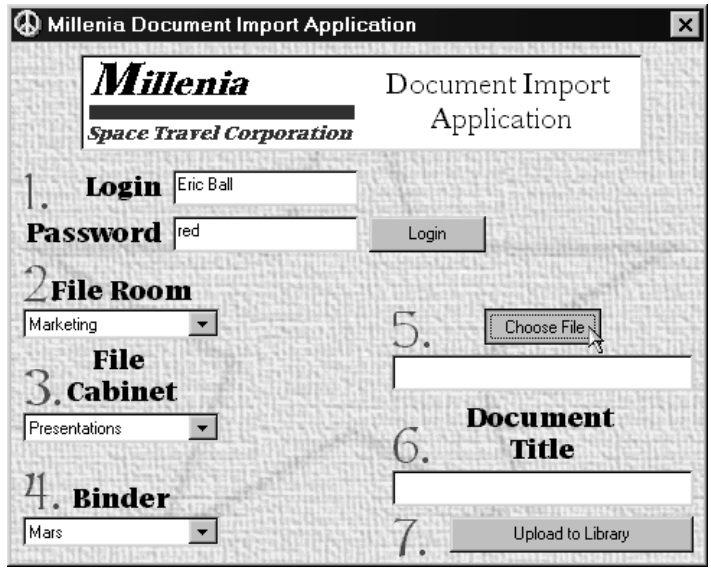
Once the file room is selected, another combo box will be populated with a list of all file cabinets that are in the specified file room and that the user has permission to see.

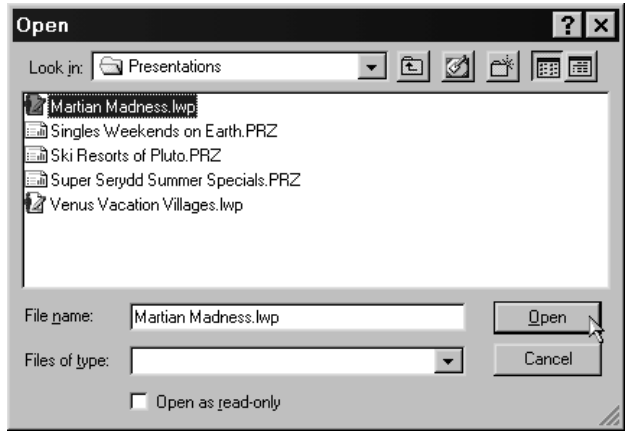


Once a file cabinet is selected, another combo box will be populated with a list of binders that are in the specified file cabinet and that the user has permission to see. The user selects a binder to which the file is to be uploaded, as shown in the following figure.

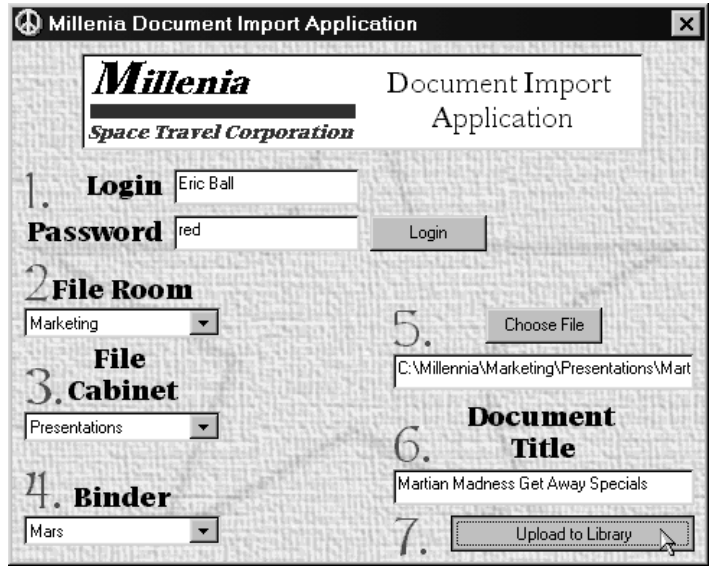


The user then either selects a file to upload (as shown in the following two figures), or types the path to a file to upload.





The user enters the document name as they want it to appear in the library (this name may be arbitrarily given by the user). This step is shown in the following figure.



The file will then be uploaded to the selected binder within the selected file cabinet, which is in turn within the selected file room.

Building the application

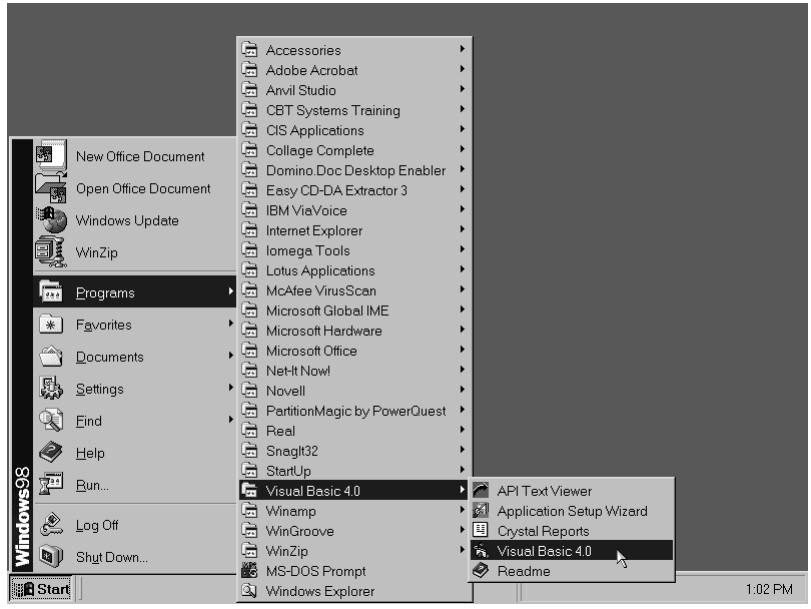
The remainder of this chapter describes the steps that were used to build the application for Millennia Space Travel Corporation using Visual Basic. These steps build a complete application that allows the user to quickly add a document to the Domino.Doc library without using a Notes client or Web browser.

Creating the Visual Basic project

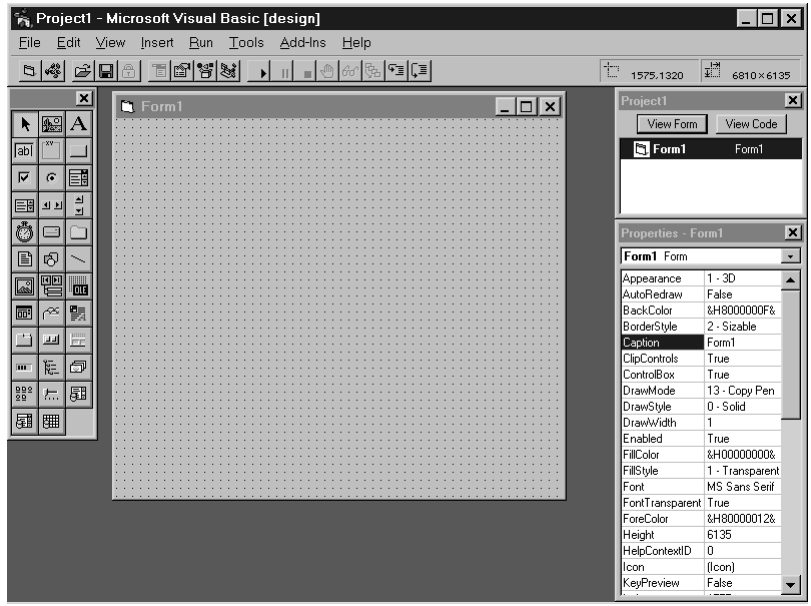
The following steps show how to prepare Visual Basic for creating the application and for using the Domino.Doc API. In this example, version 4.0 of Microsoft's Visual Basic is used (though later versions are fully supported), and version 2.5 of the Lotus Domino.Doc API is used.

1. Make sure the Domino.Doc API is installed. If you need to install the Domino.Doc API follow the steps in the section "Installing the Domino.Doc API" at the beginning of this chapter.

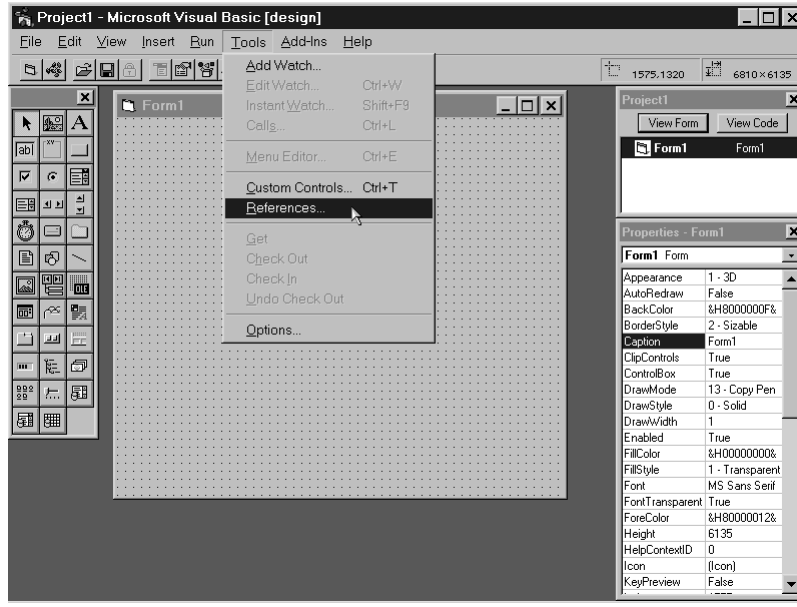
2. Start Microsoft Visual Basic, as shown in the following figure:



3. When Visual Basic opens, it will appear similar to the following figure:

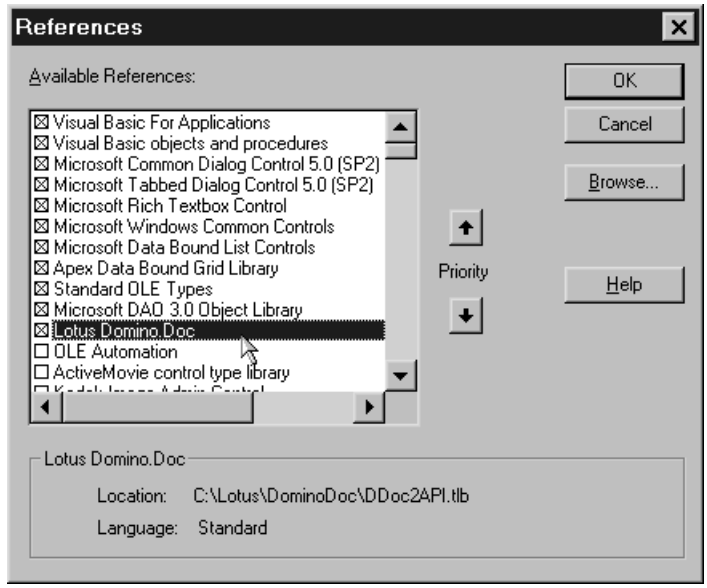


4. From the Tools menu, choose References, as shown in the following figure:



5. From the list of Available References, select Lotus Domino.Doc. In the application we are building, we do not need Lotus Notes Automation classes. If you do need to use Lotus Notes Automation classes and Domino.Doc together, one thing to remember is that both have an object class called Document, and thus you need to be careful about the priority of the references and about the definition of your object variables so that the appropriate class is used. The following figure

shows the References window with Lotus Domino.Doc selected (you may have to scroll down to find it):



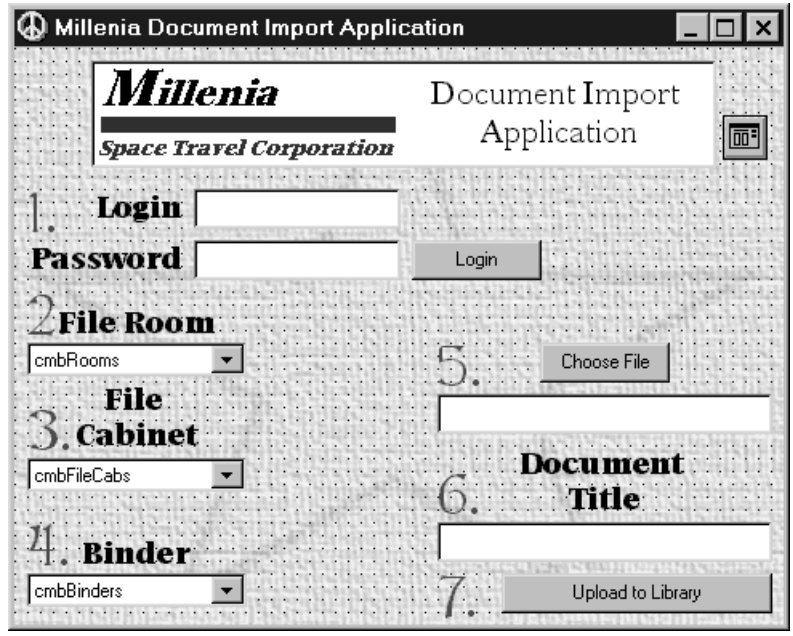
6. After selecting Lotus Domino.Doc, click OK. You are now ready to starting building the application!

Creating the form

1. Using the default frame that was created, resize it to the size that you wanted (in our example it is set to a height of 5880 and a width of 7320, which is equal to 360 by 480 pixels respectively).
2. Using the Picture property, a background image is added.
3. Create a Text Box called txtLogin. Create a Label to place beside it to label it "Login."
4. Create a Text Box called txtPassword, and create a Label to place beside it to label it "Password."
5. Create a Command Button called cmdLogin, and set the Caption property to "Login."
6. Create a Combo Box called cmbRooms, and set the Style to 2 (Dropdown List). Create a Label to place beside it to label it "File Room."
7. Create a Combo Box called cmbFileCabs, and set the Style to 2 (Dropdown List). Create a Label to place beside it to label it "File Cabinet."
8. Create a Combo Box called cmbBinders, and set the Style to 2 (Dropdown List). Create a Label to place beside it to label it "Binder."

9. Create a Common Dialog called CommonDialog1 (it doesn't matter where on the Form it is placed, as it is not seen during run-time).
10. Create a Command Button called cmdFile and set the Caption property to "Choose File."
11. Create a Text Box called txtFile.
12. Create a Text Box called txtTitle, and create a Label to place beside it to label it "Document Title."
13. Create a Command Button called cmdUp, and set the Caption property to "Upload to Library."

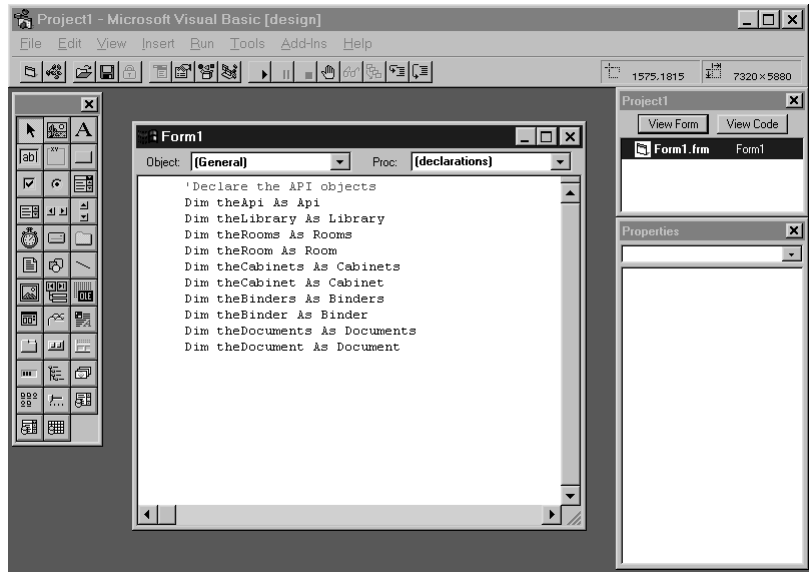
The resulting form should look similar to the following figure:



Declaring the object variables

This section shows how the object variables are globally declared. This is done using the following steps:

1. From the View menu, select code.
2. When the code window appears, choose (General) as the Object, and (declarations) as the Proc, as shown in the following figure:



3. In the code box, enter the following code (as shown in the figure above :

```
'Declare the API objects
Dim theApi As Api
Dim theLibrary As Library
Dim theRooms As Rooms
Dim theRoom As Room
Dim theCabinets As Cabinets
Dim theCabinet As Cabinet
Dim theBinders As Binders
Dim theBinder As Binder
Dim theDocuments As Documents
Dim theDocument As Document
```

Coding the cmdLogin command button

When a user clicks Login, the Click event for that Command Button runs a code that creates the API object and the Library object. A Rooms collection is taken from the Library object, and is used to fill the cmbRooms Combo Box with a list of file rooms available to the user.

To add the Login functionality, insert the following code into the cmdLogin object, Click proc code section:

```
Sub cmdLogin_Click()


      'Change the mouse cursor to an hourglass
      Screen.MousePointer = 11


      'Clear all fields (except the login and password ones)
      txtFile.Text = ""
      txtTitle.Text = ""
      Call cmbBinders.Clear
      Call cmbFileRooms.Clear
      Call cmbRooms.Clear


      'Create the Api object
      Set theApi = New Api


      'Set the login name and password
      Call theApi.SetHttpLogin(txtLogin, txtPassword)


      'Create the Library object from the Api
      '(change this for your library's URL)
      Set theLibrary = theApi.GetLibrary(
                "http://tyr.lotus.com/domdoc/milleniaLib.nsf")


      'Get the Rooms object from the Library
      Set theRooms = theLibrary.Rooms
```

```
    'Get Room count
    '     NOTE: Subtract 1 because the first object is object
    '           number 0
    iLastRoom = theRooms.Count - 1

    'Get the name of each Room object in the Rooms collection
    For i = 0 To iLastRoom
         'Get the Room object number 'i' from the Rooms
         Set theRoom = theRooms.ItemByIndex(i)
         'Add the title of the room to the Combo Box
         'called cmbRooms
         cmbRooms.AddItem theRoom.Title
    'Get the next Room object in the Rooms collection
    Next i

    'Set the mouse cursor back to normal
    Screen.MousePointer = 0

End Sub
```

Coding the cmbRooms combo box

When a user selects a file room from the list of file rooms in the cmbRooms Combo Box, a list of file cabinets available to the user from that file room populates the cmbFileCabs Combo Box. This allows the user to choose a file cabinet.

To add the functionality, the following code is used in the Click proc of the cmbRooms object:

```
Sub cmbRooms_Click()
    'Set the mouse cursor to an hourglass
    Screen.MousePointer = 11

    'Clear the other fields
    txtFile.Text = ""
```

```
txtTitle.Text = ""
Call cmbBinders.Clear
Call cmbFileCabs.Clear

'Define a variable to hold the selected file room
varRoom = cmbRooms.Text

'Get the Room object of the selected file room from
'the Rooms collection
Set theRoom = theRooms.ItemByTitle(varRoom)

'Get the Cabinets collection from the Room object
Set theCabinets = theRoom.Cabinets

'Get the Cabinet count
'     Note: Subtract 1 because the first object is object
'           number 0
iLastCabinet = theCabinets.Count - 1

'Get the name of each Cabinet in the Cabinets collection
      For i = 0 To iLastCabinet
      'Get the Cabinet object number 'i' from the Cabinets
      Set theCabinet = theCabinets.ItemByIndex(i)
      'Add the title of the file cabinet to the Combo Box
      'called cmbFileCabs
      cmbFileCabs.AddItem theCabinet.Title
'Get the next Cabinet in the Cabinets collection
Next i
```

```
    'Set the mouse cursor to normal
    Screen.MousePointer = 0

End Sub
```

Coding the cmbFileCabs combo box

After a user selects a file cabinet from the cmbFileCabs Combo Box, a list of available binders from the chosen file cabinet populates another Combo Box. To add this functionality, insert the following code into the cmbFileCabs object, Click proc code section:

```
Sub cmbFileCabs_Click()

    'Set the mouse cursor to an hourglass
    Screen.MousePointer = 11

    'Clear the list of binders
    Call cmbBinders.Clear

    'Define a variable to hold the file cabinet choice
    varFC = cmbFileCabs.Text

    'Create the Cabinet object for the selected file cabinet
    Set theCabinet = theCabinets.ItemByTitle(varFC)

    'Get the Binders collection from the Cabinet object
    Set theBinders = theCabinet.Binders

    'Get the Binder count
    '     NOTE: Subtract 1 because the first object is object
    '           number 0
    iLastBinder = theBinders.Count - 1
```

```
'Get the name of each Binder object in the Binders
'collection
For i = 0 To iLastBinder
     'Get the Binder object number 'i' from the Binders
     Set theBinder = theBinders.ItemByIndex(i)
     'Add the title of the Binder object to the Combo Box
     'called cmbBinders
     cmbBinders.AddItem theBinder.Title
'Get the next Binder object in the Binders collection
Next i

'Set the mouse cursor back to normal
Screen.MousePointer = 0

End Sub
```

Coding the cmdFile command button

After a user selects a binder from the cmbBinders Combo Box, the user may either enter a path into the txtFile Text Box, or click on the cmdFile Command Button. If the user clicks on the cmdFile Command Button, a generic file open dialog opens, and the user can browse to a desired file. The path of the file chosen is then returned to the txtFile Text Box.

To add this functionality, insert the following code into the cmdFile object, Click proc code section:

```
Private Sub cmdFile_Click()

     'Open the generic file open dialog
     CommonDialog1.ShowOpen

     'Define a variable to represent the chosen path
     varFN = CommonDialog1.filename
```

```
    'Set the txtFile Text Box equal to the chosen path
    txtFile.Text = varFN

End Sub
```

Coding the cmdUp command button

After a file is chosen, the user enters a title into the txtTitle Text Box. Then, the user clicks on the cmdUp command button to upload the chosen file to the library.

To add this functionality, insert the following code into the cmdUp object, Click proc code section:

```
Private Sub cmdUp_Click()

    'Set the mouse cursor to an hourglass
    Screen.MousePointer = 11

    'Define variables for each needed value
    varRM = cmbRooms.Text
    varFC = cmbFileCabs.Text
    varBin = cmbBinders.Text
    varFN = textFile.Text
    varTitle = txtTitle.Text

    'Create the Binder object
    Set theBinder = theCabinet.Binders(varBin)

    'Get the Documents collection from the Binder object
    Set theDocuments = theBinder.Documents

    'Create the Document object in the Documents collection
    Set theDocument = theDocuments.Add
```

```
        'Attach the chosen file to the Document object
        Call theDocument.SetContents(varFN)

        'Set the title of the Document object
        theDocument.Title = varTitle

        'Save the Document object
        Call theDocument.Save

        'Check in the Document object as a new draft, keeping
        'the old drafts
        Call theDocument.CheckIn(2, 2, False, "Created by
                        Red Book Example")

        'Set the mouse cursor back to normal
        Screen.MousePointer = 0

        'Display a Message Box telling the user it has been done
        MsgBox "The file " & varFN & " was uploaded to " & varRM
                        & " / " & varFC & " / " & varBin

        'Clear all fields so that it is ready to go again
        txtFile.Text = ""
        txtTitle.Text = ""
        txtLogin.Text = ""
        txtPassword.Text = ""
        Call cmbBinders.Clear
        Call cmbFileCabs.Clear
        Call cmbRooms.Clear

End Sub
```

This application can now be tested and then compiled. It is a fully functional custom client to upload files to a Domino.Doc library. Though this example is fairly simple, it includes all the steps necessary to create any application that interacts with a Domino.Doc library. In addition it is a fully functional and useful application.

Summary

In this chapter the Domino.Doc Application Programming Interface (API) was introduced. Some reasons for using the Domino.Doc API were discussed. The Domino.Doc API object model was discussed briefly. A custom client application that uploads files to a Domino.Doc library was then developed.

Chapter 11
Agents

This chapter defines Domino agents and Domino.Doc agents, and describes how to use them to satisfy your company's business requirements.

Millennia's Web site update notification agent is used as an example of how agents can be incorporated into the total document management picture. We also discuss the knowledge user notifier agent, an example of an agent used within the Domino.Doc product.

The following topics are covered:

- What agents are
- Agents and Security
- Steps for creating an agent
- A complete list of Domino.Doc agents
- Customized agents for Millennia Space Travel Corporation

Agents

An agent is a stand-alone program that runs in the Domino database to automate tasks. This program may affect documents in the database where it is located, other databases, the operating system, the file system, or it can interact with third-party software.

For example: You might have an agent to locate all documents that are older than 30 days, move them to another database and register this transaction in one RDBMS.

You have the following options to code the agents: Notes simple actions, Notes formulas, LotusScript, Java, or imported Java.

You can run an agent in different ways, such as using the menu bar ("manually"), by scheduling (at a certain interval), or programmatically (calling it when certain events occur). The way you invoke the agents is called the "trigger."

Agents are either private or shared. Private ones are created and used by only one user, while shared agents are created by a designer and used by anyone with the appropriate access level.

Agents and security

In this section we describe how the server document in the Domino Directory and databases' access control lists (ACL) affect the creation and running of agents.

Server document impact

An agent is able to create or delete files and databases according to the access rights level that the user or server who starts it (or who saved it for background agents) has on the server document.

The Server Access section and the Agent Restrictions section control who can perform tasks such as database creation and replication, and who can run personal, restricted, or unrestricted agents. Thus you can control the "power" of the agent in the server as follows:

Server Access section
In the Server Access section you control who can access the server, and who can create new databases and replicas.

Agent Restrictions
In the Agent Restrictions section you control who can run personal as well as who can run restricted and unrestricted LotusScript and Java agents.

By using agents written in LotusScript and Java you can include operations that have full access to the server's system and can manipulate system time, file I/O, and operating system commands.

Users or groups with unrestricted access can include more commands than others with restricted access. The unrestricted commands are those that allow access to the server's environment and file system.

Caution Unrestricted LotusScript and Java agents can potentially violate the system security.

Access Control List impact

Basically, the agent will be able to create, modify, or delete documents according to the rights that the user or server who starts it (or who saved it for background agents) has on the database.

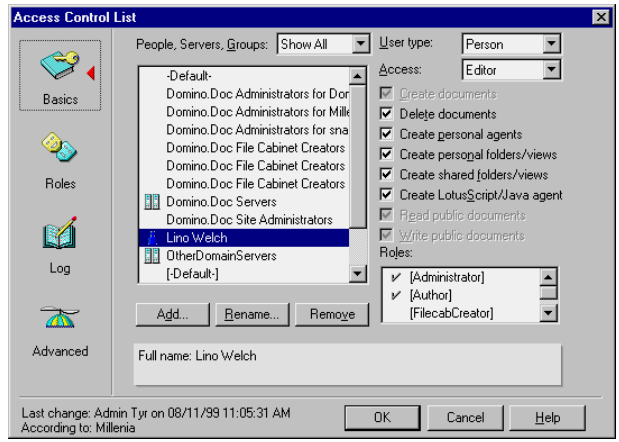
Who can create agents

The following table shows the access levels required to create agents in the database.

Agent type	*Access level required to create*
Personal	Reader or higher
Shared	Designer or higher

All the ACL refinements also affect what an agent can do in the database, but there are two of these refinements that are specifically for controlling agent creation: Create personal agents and Create LotusScript/Java.

The following figure illustrates these ACL refinements:



These ACL refinements can prevent users from creating agents and, by checking these options, the database "Administrator" (manager) can preserve a server's disk space and processing time.

Three considerations for creating your agent

After opening the database in Design mode, name your agent and specify whether it is going to be a private or a shared agent in the appropriate check box. The rest of the agent creation process is done by simply answering three important questions that define the agent.

When should this agent run? (trigger)

You need to specify when or how your agent will be started from among the following options:

- Manually From Actions Menu
- Manually From Agent List
- If New Mail Has Arrived
- Immediately Before Delivery of New Mail Document (new in Domino R5.0)
- If Documents Have Been Created or Modified
- If Documents Have Been Pasted
- On Schedule Hourly/Daily/Weekly/Monthly/Never

Which documents should it act on? (selection criteria)

This option defines on which documents your agent should act.

The options available for this field will depend on the trigger that you have chosen on the previous field. The choices are shown in the following table.

Trigger	*Selection Criteria Option*
Manually From Actions Menu	All documents in database All new and modified documents since last run All unread documents in view All documents in view Selected documents Run Once
Manually From Agent List	All documents in database All new and modified documents since last run All unread documents in view All documents in view Selected documents Run Once
If New Mail Has Arrived	Newly received mail documents
Immediately Before Delivery of New Mail Document	Each incoming mail document
If Documents Have Been Created or Modified	Newly modified documents
If Documents Have Been Pasted	Pasted documents
On Schedule Hourly, Daily, Weekly, Monthly, Never	All documents in database All new and modified documents since last run

You can refine the options above to reduce the number of documents returned by clicking the Add Search button as many times as necessary to include all appropriate selection criteria. This button is only unavailable when you choose the "Run Once" option.

The search field may be left blank when using code other than Simple Actions to write your agent because you can build the document collection using the code language.

Note Remember that Java and LotusScript agents might not follow exactly the selection criteria because you can manipulate a whole bunch of objects in the background.

What does this agent do? (code)
In this field you insert your language option, then your language code, according to the options you have chosen.

You have the following available code options:

- Simple actions
- @formulas
- LotusScript
- Java
- Imported Java

Web considerations concerning agents

WebQueryOpen and WebQuerySave are the only events available to run agents from the Web.

WebQueryOpen happens before Domino converts a document to HTML and sends it to the browser. You cannot include any outputs in this event.

WebQuerySave happens after field validation and just before saving the document.

Agents called in this event will run in the background (server side); because of this you can write your agents in LotusScript, which is a non Web-compatible language.

Tip You can also include a LotusScript Print statement to include HTML code and format your answer for Web users.

Domino.Doc agents

Domino.Doc relies heavily on agents for actions like simple navigation, Web browser functions, use of the desktop enabler, API functions when connecting via HTTP, system maintenance and so on. Scheduled agents are also responsible for database creation after thresholds are exceeded, full text index maintenance, and document backup and retrieval.

You can see a list of all the agents found in the library and file cabinet templates in Appendix D: Domino.Doc agents. Many Domino.Doc actions must be processed in the background or repetitively. These are the main reasons for the large number of agents in Domino.Doc.

Millennia's agent requirements

Millennia's customized application requires that two different agents be created: one in the Domino.Doc template in the presentation cabinet templates, and one in the Web site database template to notify people about the Web site updates.

The lifetime of a presentation, according to Millennia's policies, cannot exceed 90 days. The Domino.Doc agent notifies the marketing presentation approvers if they have a presentation that has not been modified in the last 90 days. The Web site update notification agent will run on a weekly basis and will check for the documents that have been posted on Millennia's Web site during that same week, find their categories, connect to Millennia's relational database that stores Web users' information to find their e-mail addresses, and finally notify them about the availability of new presentations that meet their interests.

Creating Millennia's agents

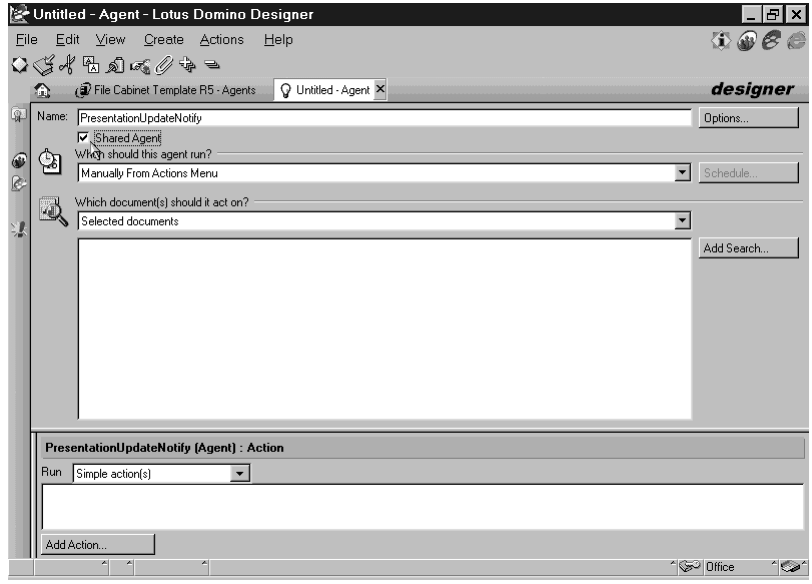
The following sections describe step-by-step how to create the two agents needed to satisfy Millennia's requirements.

Knowledge workers' notification

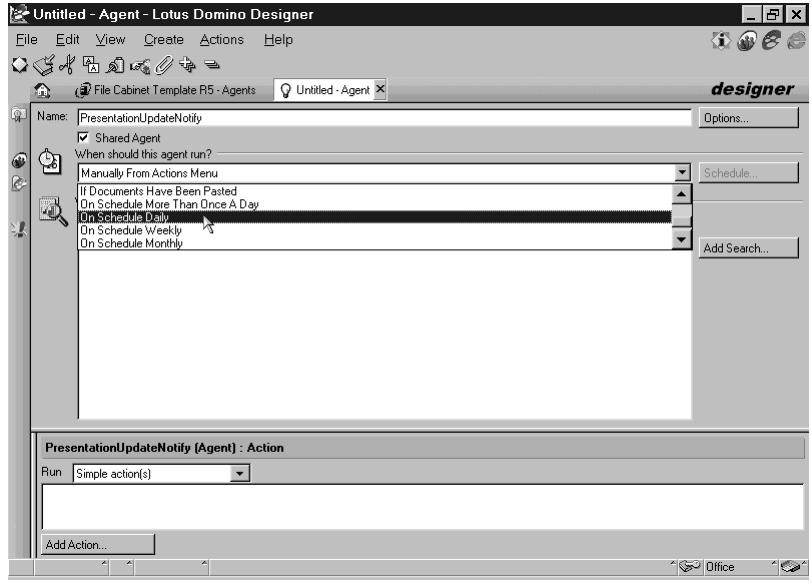
Follow these steps to create an agent that notifies the presentation approvers about outdated presentations:

1. Open the File Cabinet Template in Domino Designer.
2. Choose Create - Agent.
3. Name your agent. In our example we will call it PresentationUpdateNotify.

4. Mark it as Shared.



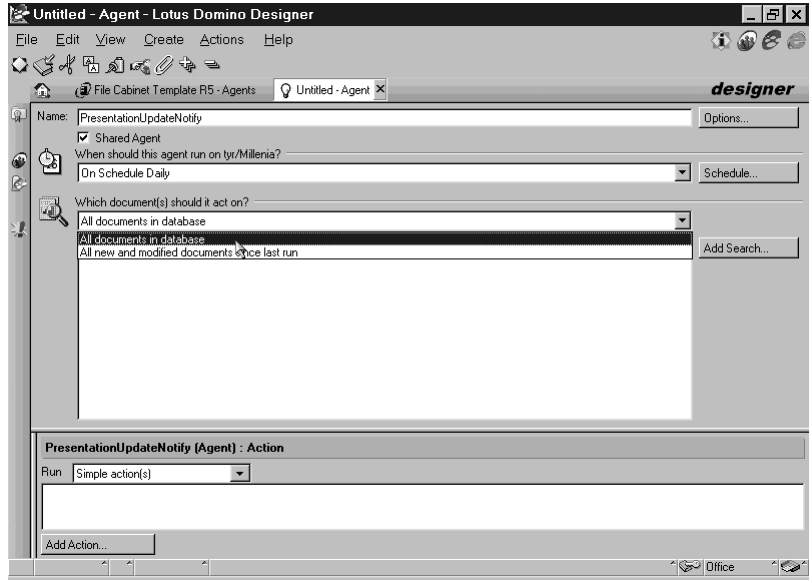
5. Choose On Schedule Daily in the When should this agent run? field.



6. Click the Schedule button and schedule your agent to run at 3:00 AM (optional).

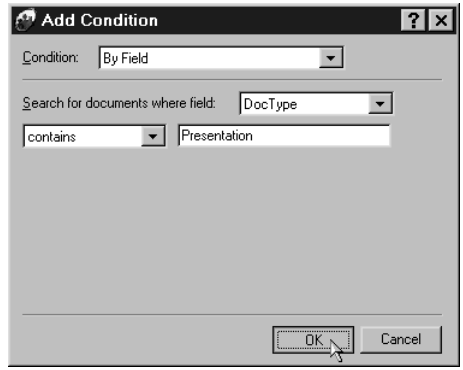
   Note You are not obligated to schedule it for 3:00 AM, but this is the time recommended since the server will be performing administrative tasks between 1:00 and 2:00 AM.

7. Choose the server that will run this agent in the Run On field.
8. Click OK.
9. Choose the All documents in database option in the Which document(s) should it act on? field.

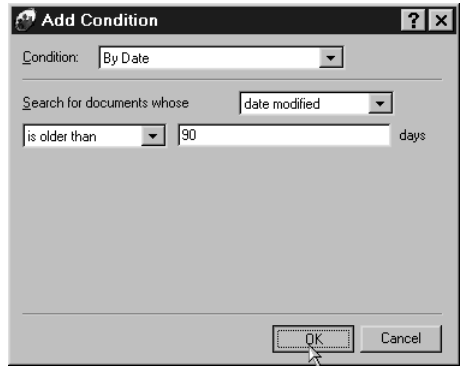


10. Click the Add Search button.
11. Choose By Field in the Condition field.
12. Select DocType in the Search for documents where field option.
13. Select the option Contains in the next field.

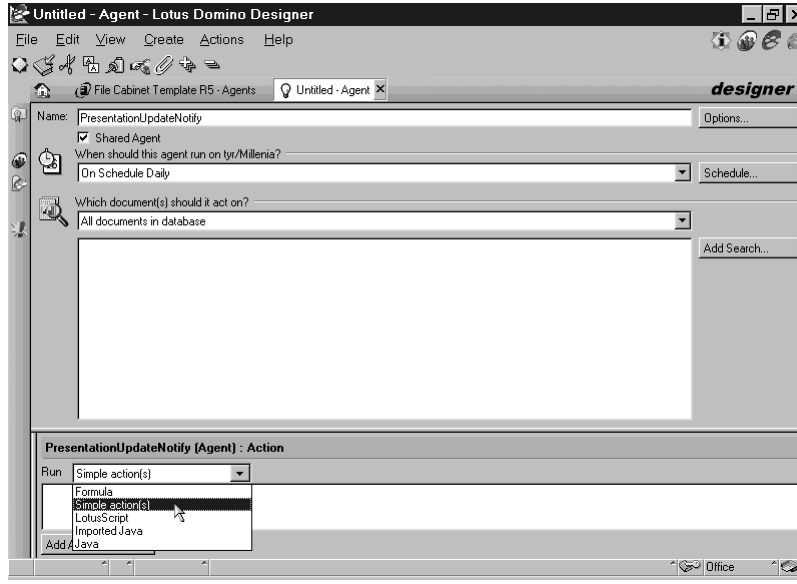
14. Type Presentation in the last field and click OK.



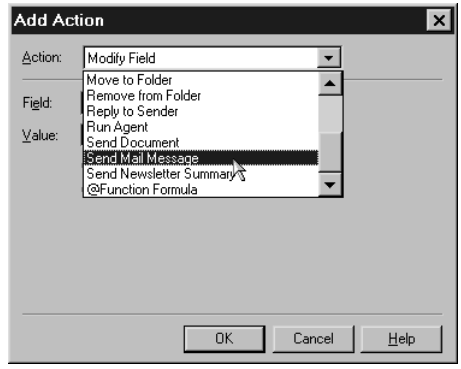
15. Click the Add Search button again to add an AND condition, then select By Date in the Condition field.
16. Select Date modified in the Search for documents whose field.
17. Select Is older than option in the next field, then type 90 in the last field and click the OK button.



18. Select Simple Actions in the Run field.



19. Click the Add Action button.
20. Select Send Mail Message in the Action field.

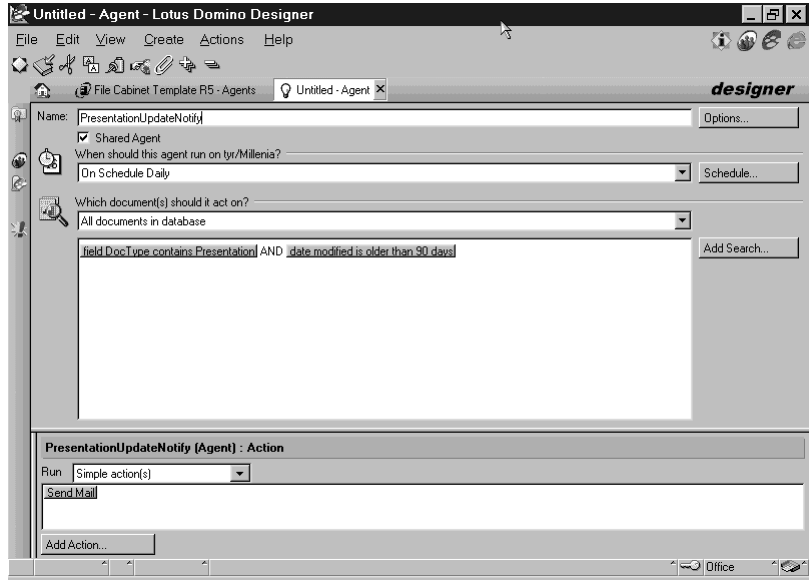


21. Select the recipients for your message.
22. Fill in the Subject and the Body fields.

23. Check the Include link to document option.

24. Choose File - Save.

Your final screen will be like the following figure when using Domino Designer R5.0:



25. Choose File - Close.

Web site update notifier

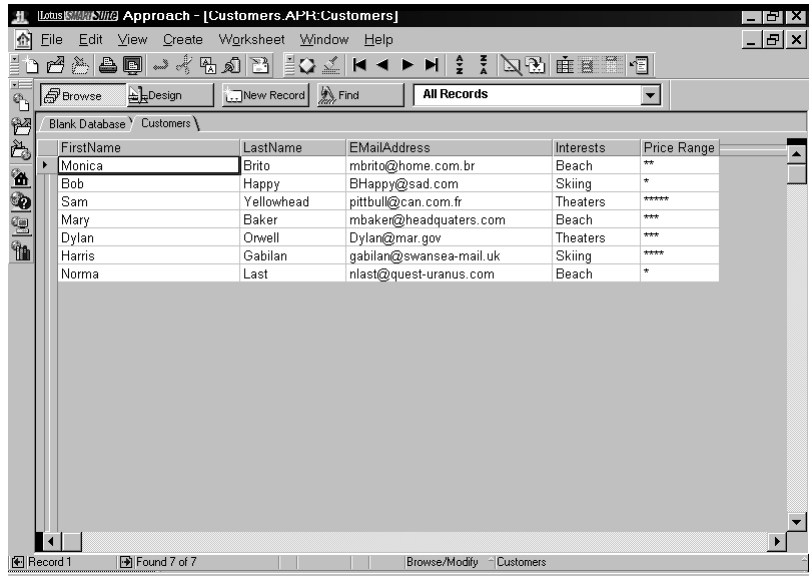
This agent should be included on Millennia's Web site in the database that contains travel presentations. The presentations are transferred to this database from Domino.Doc but it is not a Domino.Doc database itself.

This agent relies on a relational Lotus Approach database stored in the D:\Mil\Dbs subdirectory on Millennia's Domino Web server.

The customer information database was populated with information gathered from Web users.

Note If you want to try the following example you can use a new Domino database created from a blank template or you can get the one we used from the IBM Redbooks Web site. You can also get the Lotus Approach database we used from the IBM Redbooks Web site. See the appendix named Additional Web Material for instructions on how to get the files.

The following figure shows the customer database structure:



Use the following steps to create an agent that notifies Web users about site updates.

Important Be sure that you know the relational database path, name, and table name before you start writing this agent. You will be using the Open Database Connectivity (ODBC) standard to access the database. Be sure you have properly configured a System Data Source Name (DSN) entry in the Server's ODBC Datasource Administrator before you start writing your agent. For more information about using ODBC from Domino refer to Chapter 14: Using Other Database Connectivity Tools in the redbook *Lotus Domino R5.0: A Developer's Handbook*, IBM form number SG24-5331, Lotus part number CT6HPIE.

1. Open the Web site presentations database in Domino Designer.
2. Choose Create - Agent.
3. Name your agent.
4. Mark it as Shared.

5. Choose On Schedule Weekly in the When should this agent run? field.
6. Click the Schedule button and schedule your agent to run on Sundays at any time (optional).

Note You are not obligated to schedule it for Sundays, nor at a determined hour. Sunday is recommended since you generally do not have many activities on the server on this day.

7. Choose the server that will run this agent in the Run On field.
8. Click OK.
9. Choose the All documents in database option in the Which document(s) should it act on? field.
10. Select LotusScript in the Run field.
11. In the (Options) event include the following code:

```
Uselsx"*lsxodbc"
```

12. Add the following code to Initialize event:

```
' Variables declaration
Dim session As New notessession
Dim db As Notesdatabase
Dim col As Notesdocumentcollection
Dim doc As NotesDocument

' Set db to the current database
Set db=session.currentdatabase

' Build the search condition
search$= "Form=" & {"Presentation"} & _
" & @TextToTime(CreatedOn) >= " & _
"@Adjust(@Today;0;0;-7;0;0;0)"

' Perform the search in the database
Set col=db.Search (search$,Nothing,0)

' Test the search result take all the interest areas
' that have a new presentation available
```

```
' Set the results on an array eliminating the double entries
If col.count >0 Then

' Declares a dynamic array to receive the entries
     Dim catarray () As String

' Get the first document value
     Set doc = col.getfirstdocument

' Prepare the result array for the first entry
     Redim Preserve catarray(0)
     catarray (0)=doc.Attractions(0)

' Starts to loop through the whole document collection
     While Not doc Is Nothing
          exists%=0
          attr$=doc.Attractions(0)

' Check if the value already exist on the array
          For a=0 To Ubound(catarray)
               If catarray(a)=attr$ Then
                    exists%=1
               End If
          Next a

' Expand the array limit and insert the new value
     If exists%=0 Then
          Redim Preserve catarray (Ubound(catarray)+1)
          catarray(Ubound(catarray)) = attr$
     End If
```

```
' Get the next document in the document collection
                Set doc=col.GetNextDocument(doc)
           Wend

' Declare the variables to check for people in the
' Lotus Approach database
     Dim con As New  ODBCConnection
     Dim qry As New  ODBCquery
     Dim res As New  ODBCResultSet

' Connect to Approach database testing if this
' connection is made
     If con.ConnectTo("Millenia") Then

' Set then connection to be used when performing
' the query
           Set qry.connection=con

' Step through the new presentations categories using
' its entries to get people that are interest in
' this category and get their e-mail addresses
           Forall value In catarray

' Create the Query    and redefine on the next loop
' if exists
                Qry.SQL = _
"SELECT * FROM D:\Mil\Dbs\Customers.dbf WHERE Interests='" _
& value & "'"
```

```
' Define the Query to be used
                Set res.query=qry

' Execute the Query
                res.execute
' Declare a dynamic array to construct the
' mail recipients list
                Dim RecipientsList () As String

' Jump to the last row and get its number
                Call res.LastRow
                exists%=res.CurrentRow

' Test if the result has any data
                If exists%>0 Then

' Expand the array to have the same number of
' elements as rows returned
                     Redim Preserve _
 RecipientsList(exists%-1) As String

' Go back to the first row of the results
                     Call res.FirstRow

' Declare a variable and assign a value to it in order
' fill in the array and test whether its limit
' is reached
                     c%=0
' Fill in the array elements
                While c%<exists% _
  RecipientsList(c%)=res.getvalue(3)
                       Call res.nextrow()
```

```
                    c%= c%+1
                Wend

' Create a mail notification message
                Set doc = New notesdocument (db)
                doc.Form="Memo"
                doc.SendTo="Site Admin"
                doc.BlindCopyTo=RecipientsList
                Doc.Subject= _
"A new travel opportunity is waiting for you!"
                Dim msgbody As String
msgbody = _
"A new travel destination that meets your personal " &_
"interests is now available for you on our Web site." &_
Chr$(13) & Chr$(13) & "Check it now!!!" & Chr$(13) &_
"Go to Millennia's Space Travel Corporation Web site =>
http://www.lotus.com"
                Dim rtitem As NotesRichTextItem
                Set rtitem = _
New NotesRichTextItem( doc, "Body")
                Call rtitem.AppendText(msgbody)

' Submit the message
                Call doc.Send( False )

           End If

      End Forall

End If
End If
```

13. Choose File - Save.

14. Choose File - Close.

We have now created an agent that searches though all documents that have been added during the last seven days. From those documents it builds a list of attractions where new information is available. For each attraction the agent performs a lookup in the relational database to get a list of people that are interested in news about this attraction. Finally, for each of the people in the list the agent creates an e-mail message to them and sends it off.

Summary

In this chapter, we presented an overview of agents and task automation. We then discussed agent security and security considerations. We described the steps used to create an agent.

Finally, we described Millennia's requirements for several customized agents and provided the instructions for creating the agents.

Appendix A
Domino.Doc forms

This appendix lists all the forms found in the library and file cabinet templates in Domino.Doc.

Forms in the library template

The following list identifies all of the forms used by the library template.

AboutBox	InvitationNewAccount	NotesSearch
AgentArguments	InvitationWeb	ODMALaunch
BinderType	LCStatusDoc	Replica
Blank	Location	Results
BlankForm	Nav-AccessControl	Results4523
CheckOutDoc	Nav-Admin	Results46
dlgConfirmFCDelete	Nav-Blank	Revoke
dlgSelectHomeServer	Nav-Delete	Sample Notes App Integration
dlgSetFileCabPaths	Nav-EditDelete	SearchRequest
DocType	Nav-EditDeleteFA	SearchRequestV2
DownloadClient	Nav-EditDeleteGroups	SelectMasterServerDlg
DownloadComponents	Nav-EditRefresh	SystemProfile
EmptyDownloadClient	Nav-Invitation	VerifyIntegrityDlg
ErrorOrWarning	Nav-Library	WebAdvancedSearch
EvalLimit	Nav-Refresh	WebAdvancedSearchAll
FavoriteBinder	Nav-SaveCancel	WebHeader
FavoriteDoc	Nav-SaveInProgress	WebSimpleSearch
File Cabinet	Nav-Search	WebSimpleSearchAll
FileCabAccess	Nav-StartSearch	$$NavigatorTemplateDefault
FileCabLauncher	Nav-StartSearchNew	$$ReturnGeneralError
HTML	Nav-StartSearchSaved	$$ViewTemplateDefault
Invitation	Navigation	

Forms in the file cabinet template

Listed below are all of the forms used in the file cabinet template.

AboutBox	HTML	Results45
AccessTemplate	HTMLDiscovery	Results4523
Action Notification	IndexRequest	Results46
ActivityDlg	ManualArchiveDlg	Retrieval Request
ActivityTemplate	Memo	Retrieval Status
AttachFile	MultipleWIPs	Review Copy
BatchLoad	Nav-Binder	Review Expiration
Binder	Nav-Blank	RvwAprvSetupDlg
BinderTOC	Nav-Conflict	SaveRequest
Blank	Nav-ConflictReadMode	SecurityDisplay
CantRebuildFolder	Nav-CustomViews	SelectReviewCopyDlg
CheckInNewDlg	Nav-Doc	SelectUsersDlg
CheckOut Notification	Nav-DocumentViews	ServerProfile
CheckoutConflict	Nav-Main	Threshold
CheckoutOptionsDlg	Nav-Memo	ThresholdNewName
CheckoutRejection	Nav-NewDoc	TOCAdmin
CompleteApproval	Nav-Refresh	VerifyIntegrityDlg
CompleteReview	Nav-Rejection	WebApprovalComment
Completion Notification	Nav-RvwCopy	WebAttachReviewCopy
ConfirmLCCancelDlg	Nav-SaveInProgress	WebBinderHeader
CopyBinderDlg	Nav-Search	WebCheckInDocument
CopyToBinderDlg	Nav-StartSearch	WebCheckInNewDocument
CreateFolderWarning	NewBinder	WebCreateVersion
CreateRequest	NewDocument	WebDocHeader
CreateRequest4	NewProfile	WebHeader
CreateRequest5	NonMasterCheckout	WebManualArchiveDlg
CustomViews	NotesCheckInBinder	WebMulWIP
DeleteDlg	NotesCheckInDocument	WebNonMasterCheckout
Discovery	NotesCreateVersion	WebSelectBinder
DlgApprovalComment	NotesSearch	WebSelectReviewCopy
DMCheckInForm	ODMASaveFile	WebSelectUsers

continued

DocContent	PerformanceFrame	WebSetupRvwAprv
DocLink	PerformanceTest	$$NavigatorTemplate for HomeNav
Document	PleaseWait	$$ReturnGeneralError
DWF Template	ProfileInfo	$$ViewTemplate for WebCOProblems
ErrorOrWarning	ProfileInfoV3	$$ViewTemplate for WebNoFolders
GlobalProfile	ReattachArchivedFile	$$ViewTemplateDefault
HistoryTemplate	ReplicaServerDlg	(DWF Workflow Status Dialog)
HomeServerDlg	Results	(DWFRemoteLaunch)

Appendix B
Domino.Doc views

This appendix lists the views found in the library and file cabinet templates in Domino.Doc.

Library template views

View Name	*Alias*	*Purpose / Remarks*
Binder Types	vaBinderTypes	Display a list of binder types available in the library to Domino users.
Checked Out Documents NonAdmin	CODocsNA	Display a list of documents and binders checked out by the user. It is a view for ordinary users.
Checked Out Documents	CODocs	Display a list of documents and binders checked out by all users. It is a view for Domino.Doc administrators.
Document Types	vaDocTypesforNotes	Display a list of document types available in the library to Domino users.
Favorites NonAdmin	FavDocsNA	Display a list of documents and binders marked as favorite by the user. It is a view for ordinary users.
Favorites	FavDocs	Display a list of documents and binders marked as favorite by all users. It is a view for Domino.Doc administrators.

continued

View Name	*Alias*	*Purpose / Remarks*
File Cabinet Locations	vaFileCabinetLocations	Display server-specific information for each file cabinet replica, including server name, database file path, and whether or not the replica is the master. This view is maintained by Domino.Doc when replicating the library.
File Cabinet Navigation	vaFileCabinetNavigation	Display server-specific information for each file cabinet replica, including server name and the library server to access when opening the file cabinet. This view is maintained by Domino.Doc when replicating the library.
File Cabinets	vaFileCabinetsforNotes	Display a list of file cabinets that the user can access. This view is for Domino users.
Groups	vaGroups	Display a list of user groups defined for the library to Domino users.
Launch Associations	vaLaunchAssociations	Display a list of ODMA enabled application file formats defined for use with Domino.Doc.
Lifecycle Status	LCStatus	Display the life cycle status of documents you can access. This view is for Domino users.
Main View	Main View	Display a list of existing file rooms and the file cabinets they contain. This is the default view when a database is first opened.
Replica Info	vaReplicaInfo	Display a list of replicas that exist for the master library with replica server name.
(CabinetInfoView)	CabinetInfoView	Domino.Doc internal view.
(CabinetsByUNID)	CabinetsByUNID	Domino.Doc internal view.

continued

View Name	*Alias*	*Purpose / Remarks*
(CabinetSearchView)	CabinetSearchView	Domino.Doc internal view.
(CabinetsForAuthor)	CabinetsForAuthor	Domino.Doc internal view.
(ExternalCabinetLookup)		Domino.Doc internal view.
(Favorites)		Domino.Doc internal view.
(GroupsByUNID)		Domino.Doc internal view.
(HTMLLaunch)		Domino.Doc internal view.
(InterimDocuments)	InterimDocuments	Domino.Doc internal view.
(LibraryReplicas)	LibraryReplicas	Domino.Doc internal view.
(Location Info)	LocInfo	Domino.Doc internal view.
(LocationsByCabinet)		Domino.Doc internal view.
(Navigation Info)	NavInfo	Domino.Doc internal view.
(NavigationByCabinet)		Domino.Doc internal view.
(NavigationByKey)	NavKey	Domino.Doc internal view.
(NewCabinetReplicas)	NewCabinetReplicas	Domino.Doc internal view.
(ODMA Checked Out Documents)	ODMACODocs	Domino.Doc internal view.
(ODMA Favorites)	ODMAFavDoc	Domino.Doc internal view.
(SearchQueries)	SearchQueries	Domino.Doc internal view.
(SearchQueryLookup)	(SearchQueryLookup)	Domino.Doc internal view.
(StatusDocByID)		Domino.Doc internal view.
(System Administration)	SysAdmin	Domino.Doc internal view.
(ToDelete)		Domino.Doc internal view.
(Web Checked Out Documents NonAdmin)	WebCODocsNA	Display a list of documents and binders checked out by the user. It is a view for ordinary users.
(Web Checked Out Documents)	WebCODocs	Display a list of documents and binders checked out by all users. It is a view for Domino.Doc administrators.
(Web File Cabinets)	WebCabinetList	Display a list of file cabinets that the user can access. This view is for Web users.
(Web Lifecycle Status)	WebLCStatus	Display the lifecycle status of documents you can access. This view is for Web users.

continued

View Name	*Alias*	*Purpose / Remarks*
(WebBinderTypes)	WebBinderTpes	Display a list of binder types available in the library to Web users.
(WebCabinetsAccessView)	WebCabinetsAccessView	Domino.Doc internal view.
(WebCabinetsGroupView)	WebCabinetsGroupView	Domino.Doc internal view.
(WebCabinetsInvitation View)	WebCabinetsInvitation View	Domino.Doc internal view.
(WebCabinetsNew InvitationView)	WebCabinetsNew InvitationView	Domino.Doc internal view.
(WebCabinetsRegister View)	WebCabinetsRegister View	Domino.Doc internal view.
(WebCabinetsRevokeView)	WebCabinetsRevokeView	Domino.Doc internal view.
(WebDocTypes)	WebDocTypes	Display a list of document types available in the library to Web users.
(WebFavoritesNonAdmin)	WebFavDocsNA	Display a list of documents and binders marked as favorite by the user. It is a view for ordinary users.
(WebFavorites)	WebFavDocs	Display a list of documents and binders marked as favorite by all users. It is a view for Domino.Doc administrators.
(WebFileCabDocs)	WebFileCabDocs	Display a list of existing file rooms and the file cabinets they contain to Web users.
(WebSearchQueries)	WebSearchQueries	Display a list of saved library search queries in order to access the query definition of a particular search.
(WebSearchResults)	WebSearchResults	Domino.Doc internal view.

File cabinet views

View name	*Alias*	*Purpose / Remark*
(AccessView)		Domino.Doc internal view.
(ActivityView)		Domino.Doc internal view.
(All Binders)	All Binders	Display a list of binders that exist in the selected file cabinet arranged by title. This view is for Domino users.
(All By Author)	All By Author	Display a list of binders that exist in the selected file cabinet arranged by author. This view is for Domino users.
(All By Type)	All By Type	Display a list of binders that exist in the selected file cabinet arranged by type. This view is for Domino users.
(All Documents)	All Documents	Display a list of documents contained in the selected file cabinet arranged by title. This view is for Domino users.
(AllByUnid)	AllByUnid	Domino.Doc internal view.
(AllDocumentsAuthor Web No Folders)	WebNoFolders	Domino.Doc internal view.
(AllDocumentsModified Web No Folders)	WebNoFolders	Domino.Doc internal view.
(AllDocumentsTitleWeb No Folders)	WebNoFolders	Domino.Doc internal view.
(BinderIndex)	BinderIndex	Domino.Doc internal view.
(BinderLookup)	BinderLookup	Domino.Doc internal view.
(BinderProfileView)		Domino.Doc internal view.
(BinderSelect)	BinderLookup	Domino.Doc internal view.
(BinderTitleIDV2)	BinderTitleIDV2	Domino.Doc internal view.
(BinderTitleID)	BinderTitleID	Domino.Doc internal view.
(BinderTOCIndex)		Domino.Doc internal view.

continued

View name	*Alias*	*Purpose / Remark*
(BinderTOCTemplate)	BinderTOCTemplate	Used to display the binder contents. This is the default binder TOC view. This is the default design for new folders and views.
(CheckInView)		Domino.Doc internal view.
(ContentView)		Domino.Doc internal view.
(COProblemsByUNID)	COProblemsByUNID	Domino.Doc internal view.
(COProblems)	COProblems	Domino.Doc internal view.
(CreateRequests)		Domino.Doc internal view.
(DeferredCheckOutView)		Domino.Doc internal view.
(DocLookup)		Domino.Doc internal view.
(Domino.Doc)	vaNotesClient45	Domino.Doc internal view.
(DWF Archive Binder TOC Template)	DWF Archive Binder TOC Template	Domino.Doc internal view.
(DWFView)		Domino.Doc internal view.
(FileDocs)		Domino.Doc internal view.
(HistoryView)		Domino.Doc internal view.
(InterimDocuments)	InterimDocuments	Domino.Doc internal view.
(InternalAdmin)		Used to display user-created views (custom views).
(Loan Binder View)	Loan Binder View	Domino.Doc internal view.
(Nav-DocView)		Domino.Doc internal view.
(NotesBinderSelect)	BinderLookup	Domino.Doc internal view.
(Project Binder View)	Project Binder View	Domino.Doc internal view.
(SaveRequests)		Domino.Doc internal view.
(ServerProfileInfo)		Domino.Doc internal view.
(ServerProfileList)	(ServerProfileList)	Display the server profile document for file cabinets including replica server name, home server name, and whether or not the replica is the master.
(Servers)		Domino.Doc internal view.
(ToDelete)		Domino.Doc internal view.
(Trial Binder View)	Trial Binder View	Domino.Doc internal view.

continued

View name	*Alias*	*Purpose / Remark*
(VersionIndex)	VersionIndex	Domino.Doc internal view.
(Versions)		Domino.Doc internal view.
(Web All Binders)	Web All Binders	Display a list of binders that exist in the selected file cabinet arranged by title. This view is for Web users.
(Web All By Author)	Web All By Author	Display a list of binders that exist in the selected file cabinet arranged by author. This view is for Web users.
(Web All By Type)	Web All By Type	Display a list of binders that exist in the selected file cabinet arranged by type. This view is for Web users.
(WebCOProblems)	(WebCOProblems)	Display a list of check out conflict and check out rejection documents.
(WebHeaderView)		Domino.Doc internal view.
(WIPBinders)		Domino.Doc internal view.
(WIPDocs)		Domino.Doc internal view.

Appendix C
Domino.Doc navigators

This appendix lists the navigators found in the library and file cabinet templates in Domino.Doc.

Library navigators

By default, the library database contains the following navigators:

- Administration
- Library
- Replication

Administration
This navigator appears when you open the library and click the Library Administration button. It is used for administrative purposes only.

Library
This navigator is shown every time a user opens the library.

Replication navigator
This navigator is used for replication management purposes; it is viewed when a library's administrator follows these steps:

1. Open the library.
2. Click Library Administration.
3. Click Replication.

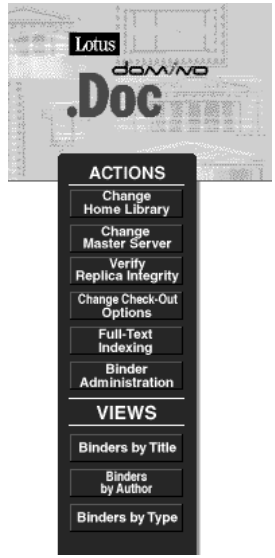
File cabinet navigators

By default, the file cabinet databases contain the following navigators:

- (Administration)
- (BinderTOC)
- (DocumentNavNext)
- (DocumentNavPrevNext)
- (DocumentNavPrev)
- (DocumentNav)
- (HomeNavforAllbyAuthorView)
- (HomeNavforAllbyTypeView)
- (HomeNav)

The (Administration) navigator

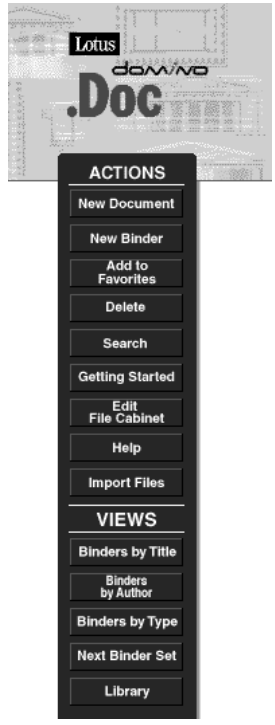
This navigator is used for file cabinet management and appears when you open the file cabinet, and then click the File Cabinet Administration button.



The (DocumentNavNext) navigator

The (DocumentNavNext) navigator is loaded when a user clicks the Documents by Title button in the (Home Nav), and only if you have more than one document storage database created by an "exceeded threshold limit" where the limit can be number of binders, documents, or database size.

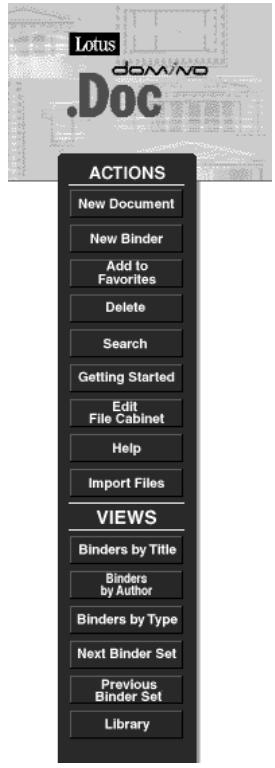
This navigator is also opened if you have more than one document storage database and you manually open the first database from your client.



The (DocumentNavPrevNext) navigator

The (DocumentNavPrevNext) navigator is loaded when a user clicks the Documents by Title button in the (Home Nav), and then clicks the Next Binder Set button. This navigator is exhibited only if you have more than two document storage databases created by an “exceeded threshold limit” where the limit can be number of binders, documents, or database size.

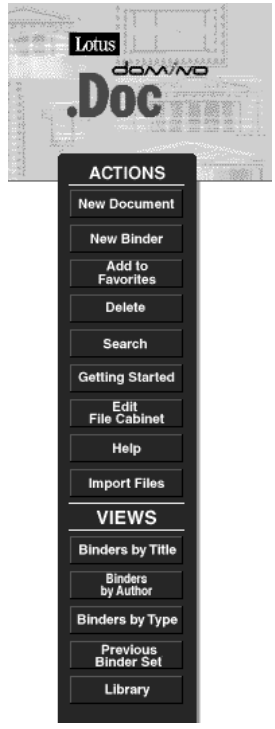
This navigator is also opened if you have more than two document storage databases and you manually opened the second database from your client.



The (DocumentNavPrev) navigator

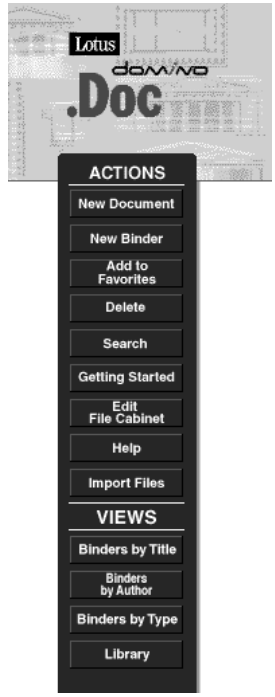
The (DocumentNavPrev) navigator is loaded when a user clicks the Documents by Title button in the (Home Nav), and then clicks the Next Binder Set button. This navigator is exhibited only if you have two document storage databases created by an "exceeded threshold limit," or if you have more than two but you are in the last document storage database.

This navigator is also opened if you have two document storage databases and you open the second database from your client, or if you have more than two document storage databases and open the last one from your client.



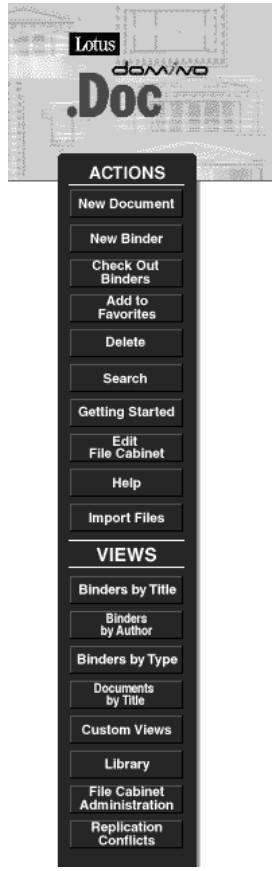
The (DocumentNav) navigator

The (DocumentNav) navigator is opened when you click the Documents by Title button on the (HomeNav) and you do not have more than one document storage database.



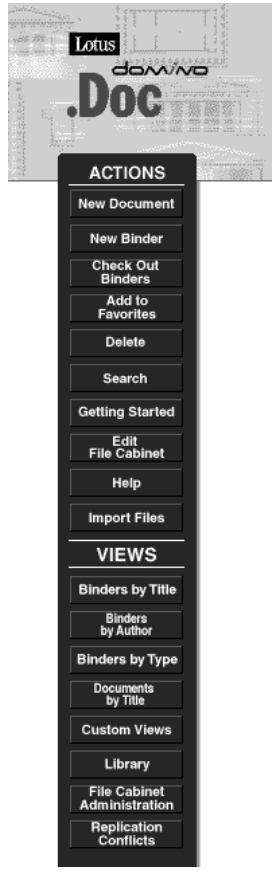
The (HomeNavforAllbyAuthorView) navigator

The (HomeNavforAllbyAuthorView) navigator is opened when you click the Binders by Author button.



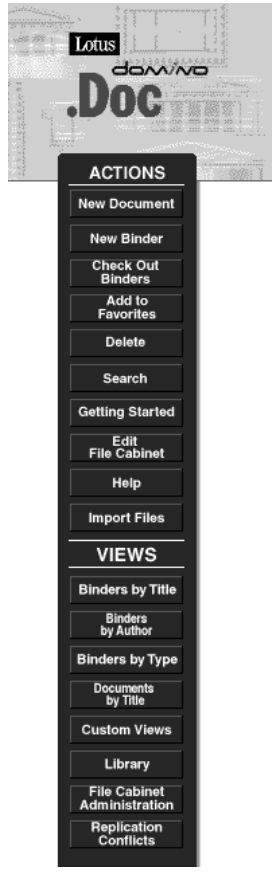
The (HomeNavforAllbyTypeView) navigator

This (HomeNavforAllbyTypeView) navigator is loaded when you click the Binders by Type button.



The (HomeNav) navigator

This is the first navigator opened when a user enters the file cabinet.



Appendix D
Domino.Doc agents

This appendix lists all the agents found in the library and file cabinet templates in Domino.Doc.

Library agents

The following table lists all the library agents.

Agent Name	*Agent Name*	*Agent Name*
Cleanup	(DMGetLibraryInfo)	(WebOnBinderType Creation)
Delete Interim Documents	(DMGetOneCabinetV2)	(WebOnDocTypeCreate)
(DMGetBinderByID)	(DMGetOneCabinet)	(WebOnFileCabinetCreate)
(DMGetCabinetsV3)	(DMProcessSearch RequestV2)	(WebOnFileCabinetDelete)
(DMGetCabinetsV4)	(DMProcessSearch Request)	(WebOnInviteNewGroup)
(DMGetCheckedOut DocsV2)	(DWF Process Setup Actions)	(WebOnItemDelete)
(DMGetCheckedOutDocs)	(MoveServer)	(WebOnRegisterUser)
(DMGetDocByID)	(NotesDeleteSearchQuery)	(WebOnRevokeAccess)
(DMGetEditableDoc)	(NotesOpenSearchQuery)	(WebOnSearchAll)
(DMGetFavoriteDocsV2)	(OnWebIssueInvitation)	(WebOnSysProfileSave)
(DMGetFavoriteDocs)	(OpenSystemProfile)	(WebPopulateQuery FormList)
(DMGetFormats)	(WebDeleteFavorite)	(WebPopulateSubformField List)

File cabinet agents

The following table lists all the file cabinet agents.

Agent Name	*Agent Name*	*Agent Name*
DocumentMonitor	(DMGetReviewCopy)	(Retrieve from BRMS)
MoveUpgradeProfile	(DMGetTOC)	(SaveAccess)
Retrieve From File System	(DMMarkForRetrieval)	(SaveDocument)
ScheduledAgents1am	(DMOpenDocument)	(SaveNewBinder)
ScheduledAgents2am	(DMProcessCreate Request4)	(SaveNewDocument)
ScheduledAgents3am	(DMProcessCreate Request5)	(Set Check Out File)
ScheduledAgentsDocs CreateorMod	DMProcessCreateRequest)	(Set Revision History)
UpgradeAgent	(DMResolveLink)	(switchuidoctoBackEnd)
(AddToBinder)	(DMSetCheckOutFile)	(TranCancelReview)
(Archive to BRMS)	(DMValidateSaveFile)	(TranCompleteRvwAprv)
(Archive to File system)	(DocumentThreshold)	(TranCreateVersion FromDraft)
(CheckInDocument ByForm)	(DWF Initiate Job Web)	(TranSendMail)
(CheckInDocument)	(DWF Initiate Job)	(TranUpdateReviewerList)
(CheckOutDocument)	(DWF Show Workflow Status)	(WebAddFavorite)
(DDMAutoLauch)	(DWFAddWorkflowStatus Web)	(WebAttachReviewCopy)
(DeferredCheckout)	(EditDocument)	(WebCancelLC)
(DelayedFolderCreate)	(EditFileCab)	(WebCompleteApproval)
(Delete Interim Documents)	(ForwardMemo)	(WebCompleteReview)
(Delete Orphaned Folders)	(Full Text Indexer)	(WebConflicts)
(DeleteDocument)	(LaunchDocument)	(WebCopyBookmark ToBinder)
(DMCheckInDocument)	(LoopBack)	(WebCreateVersion)
(DMCheckIn)	(Mark for Archive)	(WebDeleteRvwCopies)
(DMCompleteApproval)	(MarkForRetrieval)	(WebDoNothing)
(DMCompleteReview)	(NotesAddToBinder)	(WebFindTOCUNID)

continued

Agent Name	*Agent Name*	*Agent Name*
(DMDelete)	(NotesCheckOut SelectedDocs)	(WebForwardMemo)
(DMGetACL)	(NotesCopyLinkToBinder)	(WebImportReviewCopy)
(DMGetBinderACL)	(NotesProcessAprvAction)	(WebInitRvwAprvDIg)
(DMGetBindersV3)	(NotesProcess ArchiveAction)	(WebInitUsers)
(DMGetBindersV4)	(NotesProcessDocAction)	(WebLoadChildren)
(DMGetBindersV5)	(NotesProcessRvwAction)	(WebMarkForArchive)
(DMGetBinderTypes)	(ODMASaveFile)	(WebOpenLink)
(DMGetBinder WorkingCopy)	(OpenBinder)	(WebOpenMemo)
(DMGetCurrentRevision)	(OpenMemo)	(WebOpenReviewCopy)
(DMGetDocACL)	(Process Approved Documents)	(WebRejection)
(DMGetDocRevisions)	(Process Expired Reviews)	(WebSaveReviewCopy)
(DMGetDocTypes)	(ProfileInfoV2)	(WebSelectReview CopyOpen)
(DMGetDocWorkingCopy)	(ProfileInfoV3390)	(WebSelectReview CopySave)
(DMGetLatestVersion)	(ProfileInfoV3AIX)	(WebSetLCDDefaults)
(DMGetNewBinderACL)	(ProfileInfoV3AS400)	(WebSetUIDocContent)
(DMGetNewDocACL)	(ProfileInfoV3NT)	(WebSubmitApproval)
(DMGetOneBinder)	(ProfileInfoV3Solaris)	(WebSubmitReview)
(DMGetOneDocV2)	(ProfileInfo)	(WebUnmarkForArchive)
(DMGetOneDoc)	(RefreshBookmarks)	(WebUpdateLCUsers)
(DMGetProfileValuesV2)	(RequestAdvancedSearch)	
(DMGetProfileValues)	(RequestSearch)	

Appendix E
Domino.Doc object model

This appendix presents the Domino.Doc object model, detailing events, methods, and properties available in the following Domino.Doc classes:

- DocFileCabinet
- DocBinderType
- DocDocumentType
- DocReplica
- Retrieval
- SearchDisplayObject
- DocDocument

DocFileCabinet class

DocFileCabinet methods

No methods

DocFileCabinet properties

VariantArray AllowableBinderTypes

VariantArray AllowableDocumentTypes

Integer BinderThresholds*

String BinderDbReplicaID

Integer DbSizeThreshold*

Integer DefaultBinderAccess*

String DefaultBinderType

IntegerDefaultDocumentAcess*

String DefaultDocumentType

Integer DocumentThreshold

Bool EnableBinderThreshold*

Bool EnableDbSizeThreshold*

Bool EnableDocDbThreshold*

Bool EnableDocumentThreshold*

String FilePath*

Variant FileRooms*

Bool FTCaseSensitive*

Bool FTExcludeStopWordFile*

Bool FTIndexAttachments*

Integer FTIndexBreaks*

Bool FTIndexEncryptedFields*

Bool FTStopWordFile*

Bool FullTextIndex*

String GlobalBinderSubform*

StringArray GlobalBinderSubformReqFields*

String GlobalDocSubform*

StringArray GlobalDocSubformReqFields*

Integer LastError*

String LastErrorMessage*

String MasterServer

String NavServer

StringArray ReplicaServer

String Server

String Title*

DocFileCabinet events

PostDeleteFileCabinet

PostSaveFileCabinet

QueryDeleteFileCabinet

QuerySaveFileCabinet

DocBinderType class

DocBinderType methods

No methods

DocBinderType properties

Integer LastError*

String LastErrorMessage*

DocBinderType events

PostDeleteBinderType

PostSaveBinderType

QueryDeleteBinderType

QuerySaveBinderType

DocDocumentType class

DocDocumentType methods

No methods

DocDocumentType properties

Integer LastError*

String LastErrorMessage*

DocDocumentType events

PostDeleteDocumentType

PostSaveDocumentType

QueryDeleteDocumentType

QuerySaveDocumentType

DocReplica class

DocReplica methods

No methods

DocReplica properties

Integer LastError*

String LastErrorMessage*

DocReplica events

PostSaveReplica

QuerySaveReplica

Retrieval class

Retrieval methods

bool= CanRetrieve

Retrieval properties

DocDocument Document

Integer LastError*

String LastErrorMessage*

Bool ShouldNotify

Retrieval events

No events

SearchDisplayObject class

SearchDisplayObject methods

New()

SearchDisplayObject properties

Integer LastError*

String LastErrorMessage*

String NavigatorForm*

Variant ObjectContext*

String QueryString

SearchDisplayObject events

SearchEndDisplay

SearchEndDisplayFileCabResults

SearchProcessResults

SearchStartDisplay

SearchStartDisplayFileCabResults

DocDocument class

DocDocument methods

String ArchiveFileName

AttachArchiveFile(InputFile$)

AttachSourceFile(SourceFile$)

AttachViewFile(InputFile$)

Bool CanArchive

NotesRichtextItem = CreateAbstract

ExtractArchiveFile(DestFile$,Force)

ExtractSourceFile(DestFile$,Force)

ExtractViewFile(DestFile$,Force)

Variant GetFieldValue(ItemName$)

Bool HasArchiveFile

Bool HasSourceFile

Bool HasViewFile

LogActivityEntry (Message$)

LogMessage(Message$,Ncode%,Nseverity%,Module$)

MarkForRetrieval

RemoveAbstract

RemoveArchiveFile

RemoveSourceFile

RemoveViewFile

SaveMethod

SetFieldValue(ItemName$,Value)

String ViewFileName

DocDocument properties

NotesRichTextItem Abstract

Bool AllDocEditors*

Bool AllEditors*

Bool AllReaders*

Bool Approvers *

Bool AprvAllDocManagers*

Bool AprvAllDraftEditors*

StringConstant AprvCompleteAction*

Integer AprvExpireTime*

String AprvMessage*

StringConstant AprvNotifyAfter*

Bool AprvOnExpireNotifyApprover*

Bool AprvOnExpireNotifyOriginator*

StringConstant AprvRouting*

String ArchiveInfo*

Integer ArchiveState*

String CheckOutUser

String CreateAuthor

NotesDateTime CreateDate

StringArray DocEditors*

StringDocFileName

String DocFormat

String DocID

StringArray DocManagers*

StringArray DocReaders*

String DocType*

NotesDocument Document

StringArray DraftEditors*

String ID

Integer IsVersion

Bool IsWorkingCopy

Integer LastError*

String LastErrorMessage*

String LastModAuthor

NotesDateTime LastModDate

String LifeCycleState

String ParentBinderType

String ParentBinderTitle

StringArray Reviewers*

Bool RvwAllDocManagers*

Bool RvwAllDraftEditors*

Bool RvwEditContent*

Integer RvwExpireTime*

String RvwMessage*

StringConstant RvwNotifyAfter*

Bool RvwOnExpireNotifyOriginator*

Bool RvwOnExpireNotifyReviewer*

StringConstant RvwRouting*

String Title*

NotesDocument TypeInfo

String Version

DocDocument events

PostAddToBinder

PostApprovalComplete

PostCheckIn

PostCheckOut

PostDeleteDocument

PostReviewComplete

QueryAddToBinder

QueryCheckIn

QueryCheckOut

QuerySetupApproval

QuerySetupReview

QuerySubmitApproval

QuerySubmitReview

QueryDeleteDocument

Appendix F
Domino.Doc API objects and classes

This appendix contains details about the Domino.Doc API objects and classes.

Collection classes

The following table lists the common collection class properties and methods:

Property or method	*Description*	*Valid for*
Count	Returns the number of items in the collection	All collection classes
ItemByIndex	Accesses an object by its position in a collection	All collection classes
ItemByTitle	Accesses an object in a collection using the Title property	Rooms, Cabinets, Binders, Documents
ItemByName	Accesses an object in a collection using the Name property	Formats, Profiles, Fields, Keywords, Users
ItemByValue	Accesses an object in a collection using the Value property	Keywords
Item	Accesses an object in a collection using a variant with either a numeric or string value	All collection classes
IsCached	Indicates cache status of the collection	All collection classes except Keywords and Users
IsValid	Indicates if object is valid to use, based on cache status of parent or ancestor object	All collection classes
Cache	Explicitly stores collection data on the client	All collection classes except Keywords and Users
Uncache	Releases stored objects on the client	All collection classes except Keywords and Users
Add	Adds a new item to the end of a collection	Binders and Documents

Room objects

The following table lists the properties of a Room object:

Property	*Data Type*	*Description*
Cabinets	Cabinets	Cabinets collection contained in a room
IsValid	Boolean	Validity of object based on cache status of the parent object
Library	Library	Parent library of a room
Title	String	Descriptive title of a room

Cabinet objects

The following table lists the Cabinet object class properties:

Property	*Data Type*	*Description*
BinderDatabase	String	Replica id of the binder database
BinderProfiles	Profiles	List of profiles for binders in a file cabinet
BinderSecurity	Security	Default binder security for binders created in a file cabinet
Binders	Binders	Binders collection contained in a file cabinet
DocumentProfiles	Profiles	List of profiles for documents in a file cabinet
Id	String	Unique ID of a file cabinet
IsMaster	Boolean	Master or replica file cabinet
IsValid	Boolean	Validity of object based on cache status of its parent object
Library	Library	Parent library of the cabinet
MasterServer	String	Name of the server that contains the master replica of a file cabinet
NotesServer	String	Name of the Domino server where a file cabinet resides
Rooms	Rooms	List of rooms to which a file cabinet belongs
Title	String	Descriptive title of a file cabinet
UniversalId	String	Universal ID of the file cabinet database

Cabinet objects

The following table lists the Cabinet object class methods:

Method	*Description*
Cache	Gets cabinet object from server and stores it in this cabinet object
Clone	Clones cabinet object
SearchDocuments	Searches the cabinet and returns a collection of documents based on search criteria

Binder objects

The following table lists the Binder object class properties:

Property	*Data Type*	*Description*
Cabinet	Cabinet	Parent cabinet of the binder
DocumentDatabase	String	Replica Id of database
DocumentSecurity	String	Default document security for documents created in a binder
Documents	Documents	Collection of document objects in the binder
Id	String	Unique ID of a binder
IsCabinetReadOnly	Boolean	True if binder's cabinet contents are read-only
IsValid	Boolean	Validity of object based on cache status of its parent
Library	Library	Parent library of the binder object
Profile	Profile	Set of fields that define attributes of the binder
Security	Security	User access control on the binder
UniversalId	String	Universal ID of the binder
Title	String	Descriptive title for binder

The following table lists the Binder object class methods:

Method	*Description*
Cache	Gets binder from server and stores it in this binder object
CheckIn	Releases new or revised binder to server
Clone	Creates copy of binder object
Delete	Removes empty binder from cabinet
Save	Saves binder object to server

Document objects

The following table lists the Document object class properties:

Property	*Data Type*	*Description*
Binder	Binder	Parent binder for the document object
CabinetTitle	String	Title of cabinet containing the document
CheckedOutDate	String	If document is checked out, indicates the date it was checked out
CheckedOutUserName	String	If document is checked out, contains the name of the user who checked it out
CreationDate	String	Date the document was created
DraftDocument	Document	The latest draft of the current document
DraftNumber	Long	Draft number of the current document
DraftOption	Long	User's options with respect to checking in the current document as a draft (1=must check in as new, 2= may check in as new, 3=must check in as replacement, 4=drafts not allowed)
Extension	String	File extension of the document's file attachment
FileName	String	Name of the document's file attachment
HasBinder	Boolean	True if the current Document object has a valid parent Binder object
HasDraft	Boolean	True if the current Document object has a draft copy
HasReviewCopy	Boolean	True if the current Document object has a review copy

continued

Property	*Data Type*	*Description*
HasVersion	Boolean	True if the current Document object has a version copy
Id	String	Unique ID assigned to the document from Domino.Doc
IsApprover	Boolean	True if the user is an approver
IsArchived	Boolean	True if document is archived
IsCurrentRevision	Boolean	True if the current copy of the current document is the latest version or draft
IsLink	Boolean	True if the current document is a link
IsLinkBookmark	Boolean	True if document link is a bookmark
IsLinkCheckedOut	Boolean	True if document is a checked-out document
IsLinkFavorite	Boolean	True if document link is a favorite document
IsLocked	Boolean	True if document is checked out or archived
IsReviewCopy	Boolean	True if the current copy of the current document is the review copy
IsReviewer	Boolean	True if the user is a reviewer
IsValid	Boolean	Validity of the Document object, based on the cache status of its parent object
IsVersion	Boolean	True if the document is a version, false if the document is a draft
IsWorkingCopy	Boolean	True if the current copy of the current document is the working copy
Library	Library	Parent Library object of the current Document object
LifeCycleState	Long	Current stage of the life cycle of the document (1=new, 2=draft, 3=in review, 4=review complete, 5=pending approval, 6=approved, 7=rejected, 8=released)
LockState	Long	Document lock state that determines if a document can be checked out (1=checked out, 2=provisional, 3=rejected, 4=conflict)
Profile	Profile	Set of fields that define attributes of the document
Security	Security	User access control on the document
Title	String	Descriptive title of document
UniversalId	String	The universal ID of the current document

continued

Property	*Data Type*	*Description*
VersionDocument	Document	The version of the current document
Version number	Long	Version number of current document
VersionOption	Long	User's option with respect to checking in as a version

The following table lists the Document object class methods:

Method	*Description*
Approve	Approves a document that is pending approval
CheckIn	Checks in, to the server, a working copy
Clone	Returns a new object reference to the current document
CompleteReview	Completes the user's review of a document that is in review
DeleteAllRevisions	Deletes all versions and drafts of the current document
GetContents	Downloads the current document's file attachment
MarkForRetrieval	Marks an archived document for retrieval
Reject	Rejects the document if it is pending approval
ResolveLink	Returns the actual document referenced by the link
Save	Saves, to the server, changes made to a working copy
SetCheckedOutFileName	Changes the pathname of the file attachment of the working copy
SetContents	Uploads a file to working copy or review copy document
SetToReviewCopy	Sets the document object to review copy
SetToWorkingCopy	Sets the document object to working copy, checking it out if it is not already checked out

Security objects

The following are the properties of Security objects:

Property	*Data Type*	*Description*
allEditorsFlag	Boolean	Editor access inheritance from parent cabinet or binder
allReadersFlag	Boolean	Reader access inheritance from parent cabinet or binder
IsCached	Boolean	Cache status of this Security object
IsSecured	Boolean	Indicator of binder security. Has no meaning for document objects
IsValid	Boolean	Validity of object based on cache status of parent
Library	Library	Parent library of the security object
Users	String	Users collection in the security object

The following are the Security object methods:

Method	*Description*
Cache	Caches security object
Uncache	Uncaches security object

User objects

The following are the User object properties:

Property	*Data Type*	*Description*
Default	Long	Default access level for the current user
DefaultIfFlags	Boolean	Default access level for the current user, given the flags settings specified in the parameters
IsValid	Boolean	Validity of object based on cache status of parent
Library	Library	Parent library of the user object
Maximum	Long	Maximum allowable access level for this user for objects in the current cabinet or binder
Minimum	Long	Minimum allowable access level for this user for objects in the current cabinet or binder
Name	String	User name
Security	Security	Security object for the current user
Value	Integer	Access level for the current user for this object

Note The security levels are defined as follows:

- 3 Manager
- 2 Editor
- 1 Reader
- 0 No access

Profile objects

The following are the Profile object properties:

Property	*Data Type*	*Description*
Fields	Fields	Fields collection in the profile
IsValid	Boolean	Validity of object based on cache status of parent
Library	Library	Parent library of the profile object
Name	String	Descriptive name of profile

Field objects

The following are the Field object properties:

Property	*Data Type*	*Description*
DefaultValue	String	Default value for field
IsRequired	Boolean	Whether field requires a value
IsValid	Boolean	Validity of object based on cache status of parent
Keywords	Keywords	Keywords collection in the field
Library	Library	Parent library of the field object
Name	String	Descriptive name of field element
Profile	Profile	Parent profile of the field
RequiresRefresh	Boolean	Whether refresh is required when value changes
Value	String	Value of field element

Keyword object

The following are the Keyword object properties:

Property	*Data Type*	*Description*
Field	Field	Parent field of the keyword
IsValid	Boolean	Validity of object based on cache status of parent
Name	String	Text of the keyword element
Value	String	Value associated with the keyword element

Appendix G
ODMA and the Desktop Enabler

This chapter introduces the topic of the Open Document Management API standard, and describes how it relates to Domino.Doc. The Domino.Doc Desktop Enabler is also discussed.

Introduction

The Open Document Management API (ODMA) is a standardized, high level interface between desktop applications and document management systems. ODMA is a vendor-independent standard that:

- Allows seamless integration of a document management system so that it appears to the user as an extension of the desktop applications
- Allows desktop application vendors to create applications that can easily integrate all ODMA-compliant document management systems
- Allows document management system vendors to create systems that can easily integrate with all ODMA-compliant desktop applications
- Reduces the effort needed to create and maintain document management solutions

Document management systems are most effective when integrated seamlessly with the applications being used by the end-users, whether office automation software (such as Lotus SmartSuite or Microsoft Office), or highly specialized applications (such as computer aided drafting and design applications or laboratory information systems).

In July of 1994, the first version of the ODMA standard was released. ODMA 1.0 consisted of two parts:

- A platform-independent API
- Platform-specific registration and binding specifications

Since then, many companies have either shipped products that are ODMA 1.0 compliant or have announced that they will soon be releasing products that are compliant.

ODMA 2.0 will include Workflow and Win32 compliance in the standard. Future implementation will include multiple platforms, including UNIX and Macintosh.

Domino.Doc and ODMA-enabled applications

Domino.Doc uses facilities built into a set of applications that support the ODMA specification. Applications (both desktop and document management systems) that are ODMA compliant are called ODMA-enabled applications. When using an ODMA-enabled application, you can work in an application-centric mode of operation where you access Domino.Doc directly from your desktop application rather than using a Notes client or browser to access Domino.Doc documents.

Domino.Doc includes an ODMA-enabled interface called the Domino.Doc Desktop Enabler. When using the Desktop Enabler, Domino.Doc takes over the ODMA-enabled desktop applications' File - Open, File - Save, File - Save As, and File - Close menu items. When you choose these tasks you do not see the common Windows dialog boxes normally used in the applications to access the file system, but rather you get direct access to the Domino.Doc library. This allows you to create, access, or save your documents without ever leaving the familiar desktop application.

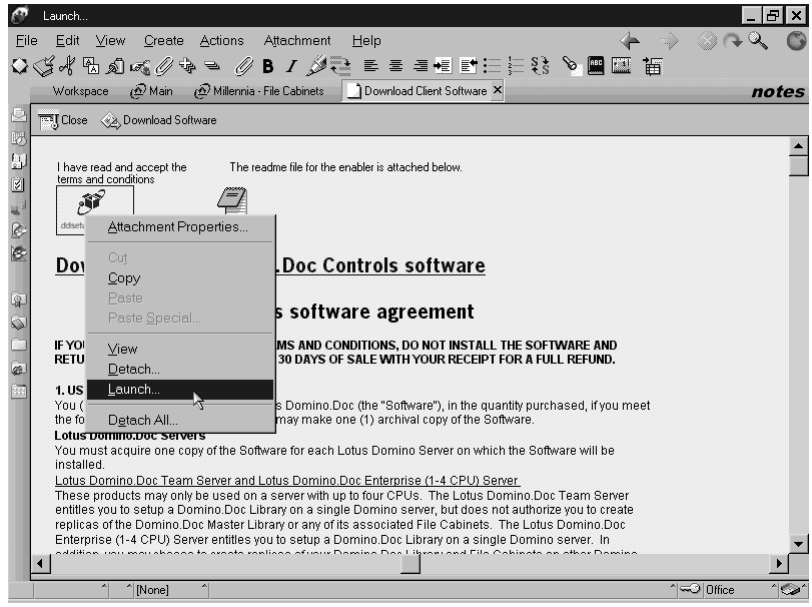
The ODMA support provided by Domino.Doc facilitates access to Domino.Doc during a single application work session in which you open, check out, edit, and check in an existing document; or create, save, and check in a new document. It also provides the functionality to edit a document in multiple work sessions, by saving intermediate changes to the Domino.Doc library and allowing you to access your work in progress during another session.

Using the Desktop Enabler

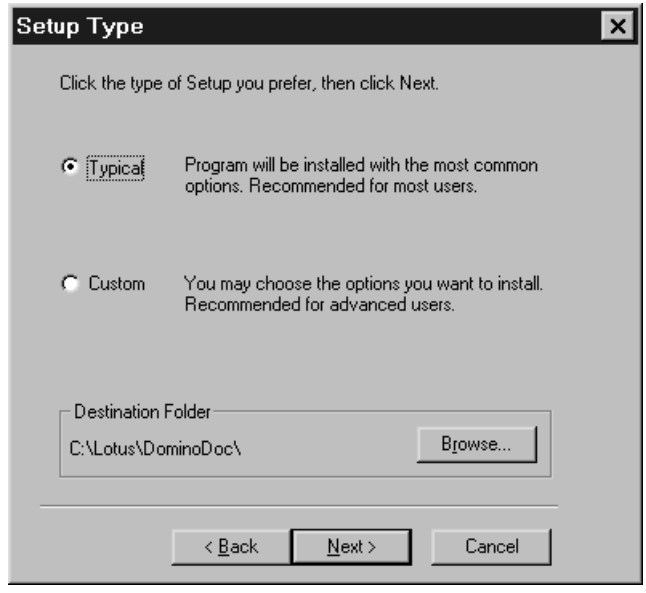
This section outlines how to use the Domino.Doc Desktop Enabler to integrate Domino.Doc with other desktop applications.

Installing the Desktop Enabler

To install the Domino.Doc Enabler, enter the library and choose Library Administration. When the Library Administration navigator appears, click Download Client Software. A document displaying the Lotus software agreement will be opened. Near the top of the page there is an icon representing an attached file called ddsetup.exe. Right click the icon, and choose launch (as shown in the following diagram).



After this begins running, some information dialogs will be displayed. Simply click Next to get past them. When asked if you want a Typical installation or a Custom installation, choose Typical (as shown in the following figure). Click Next, and follow the prompts on your screen.



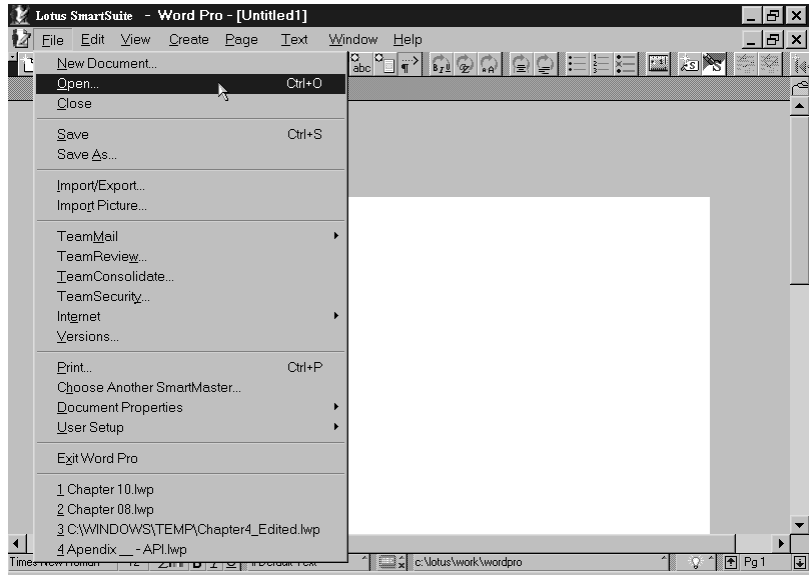
Using the Desktop Enabler within applications

The Domino.Doc Desktop Enabler changes the functionality of the File - Open, File - Save, File - Save As, and File - Close menu choices. Instead of the usual common dialog boxes for open or save, your screen will display a dialog box that provides direct access to the Domino.Doc library.

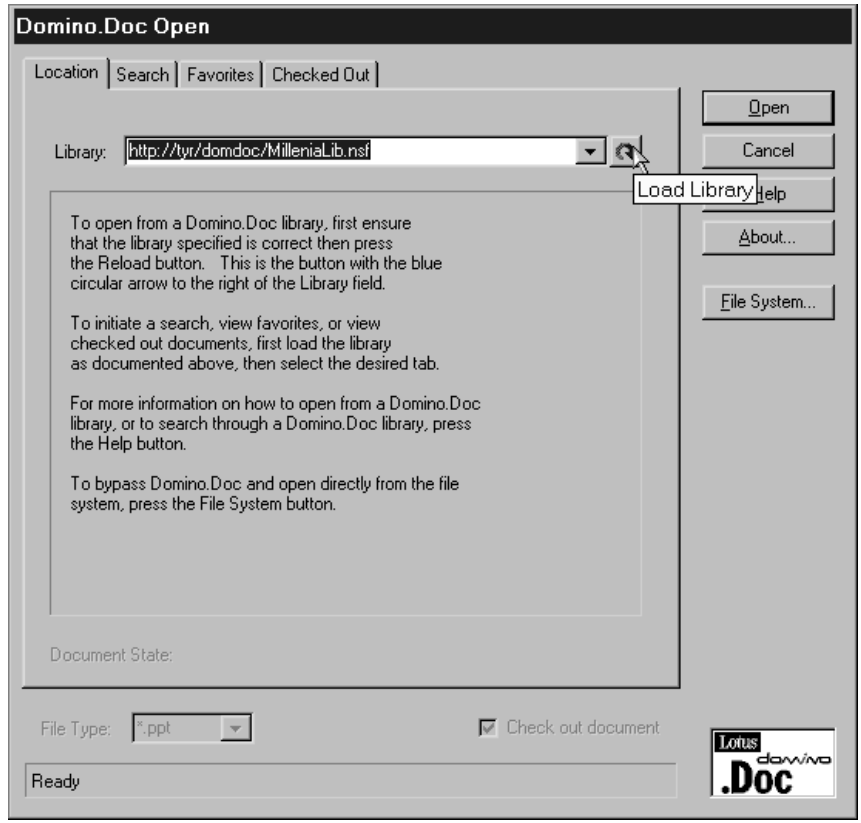
Opening a document

In the following example, a document is opened from an ODMA-enabled desktop application (in this case Lotus Word Pro), and is intercepted by Domino.Doc (an ODMA-enabled document management system).

1. From the File menu, choose Open (as shown in the following figure):



2. The Domino.Doc Desktop Enabler dialog box appears, as shown in the following figure:



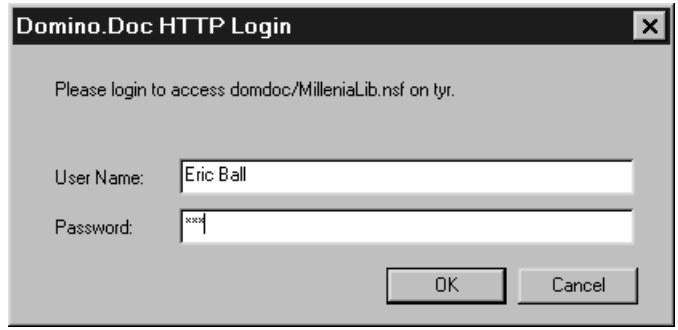
3. Enter the URL of your library (either the Notes protocol or the Hyper Text Transfer Protocol can be used by preceding the address with *Notes:* or *HTTP:* respectively).

   Note If you are using Notes protocol and must specify a hierarchical name for the server (like "Tyr/Millenia") the server name and the library file path should be separated by two exclamation points ("!!"). Here is an example:

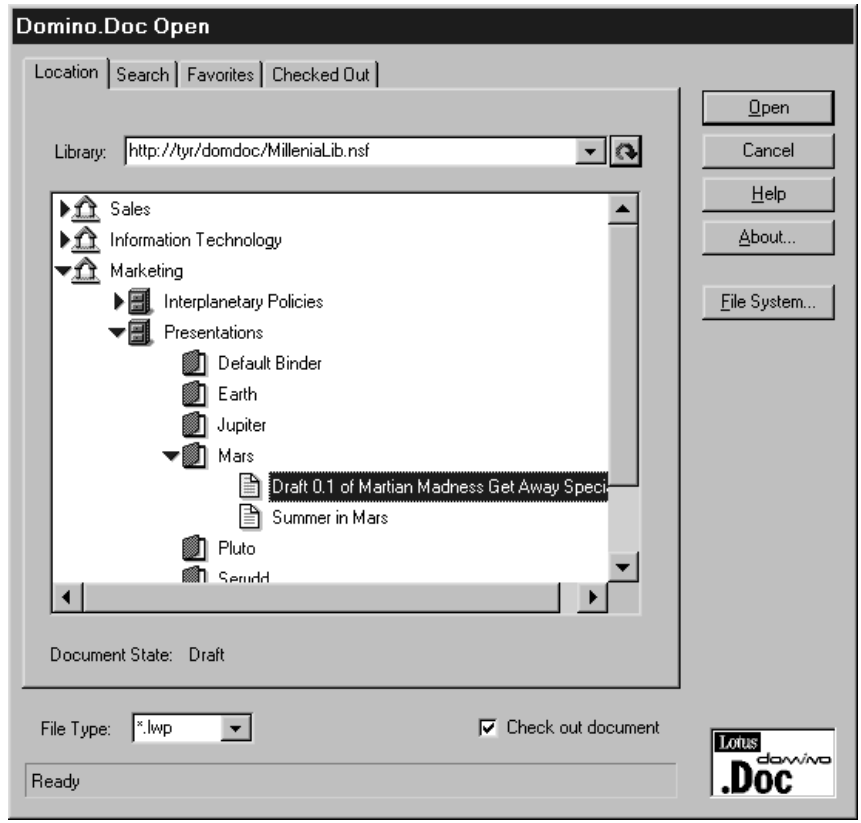
   `Notes://Tyr/Millenia!!domdoc/Millenialib.nsf`

4. Click the round arrow icon to load the library (as shown in the previous figure).

5. You will be prompted for either a password (if you used the Notes protocol) or a user name and password (if you used HTTP), as shown in the following figure:



6. Browse through the file rooms, file cabinets, and binders to get the document you desire (as shown in the figure below):



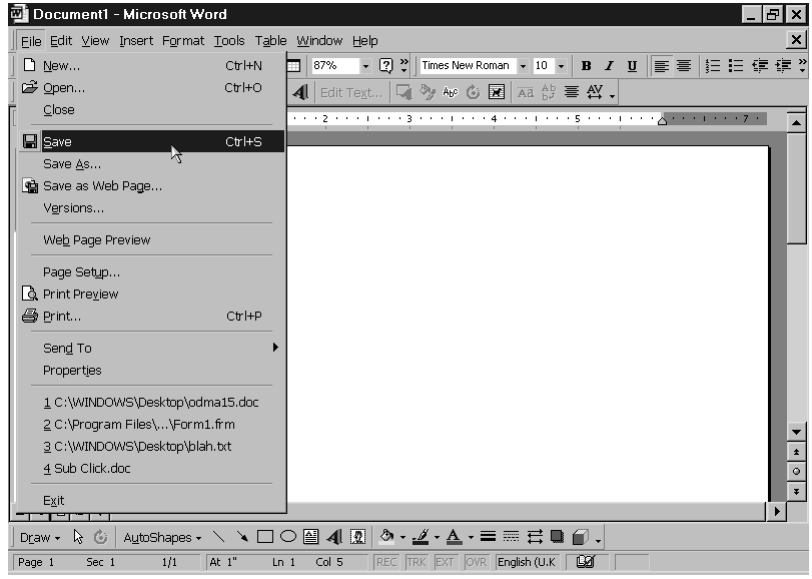
Note You can only see files of the selected file type. In the above figure we have chosen only to see Lotus Word Pro files (*.lwp). If you can not find a file that should be in the library check to see whether you are using the correct file type filter.

7. Click OK to open the document.

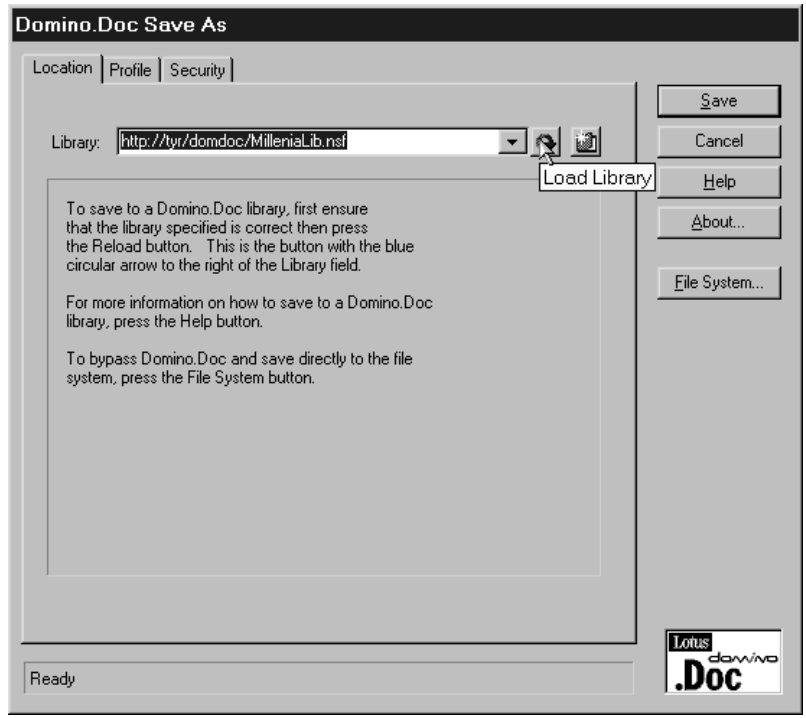
Saving a document

This example shows saving a document from an ODMA-enabled application (in this case Microsoft Word) to Domino.Doc.

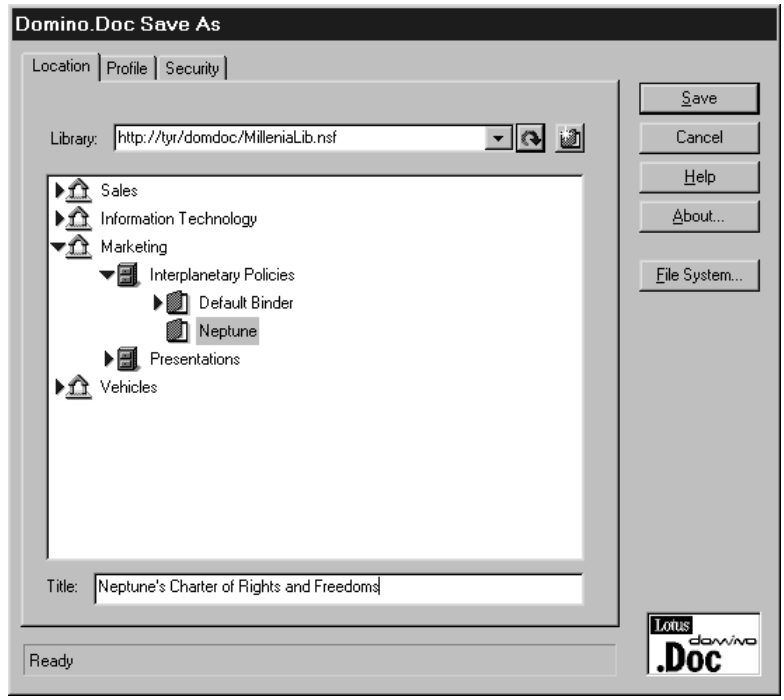
1. From the File menu, choose Save.



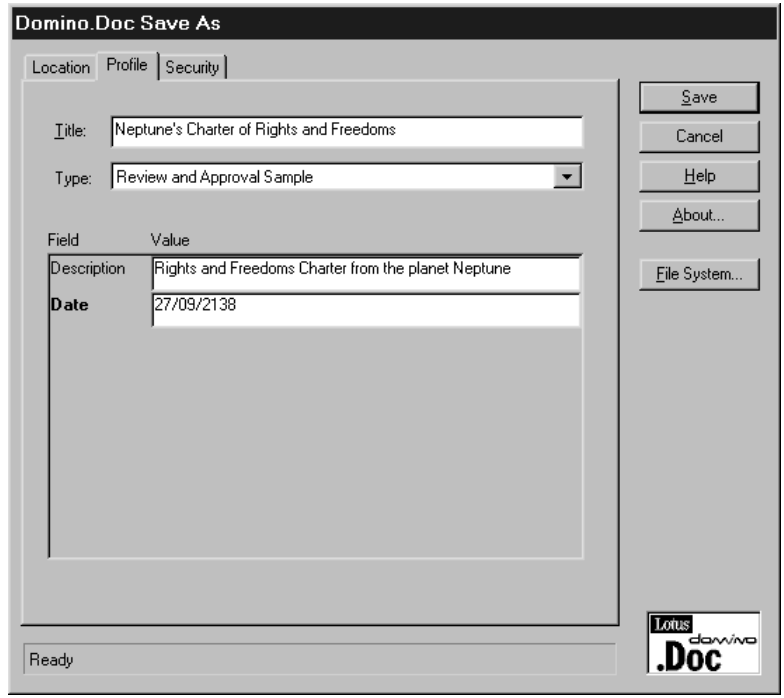
2. Enter the URL of the library (either using "Notes:" or "HTTP:" preceding it, depending on which protocol is to be used).



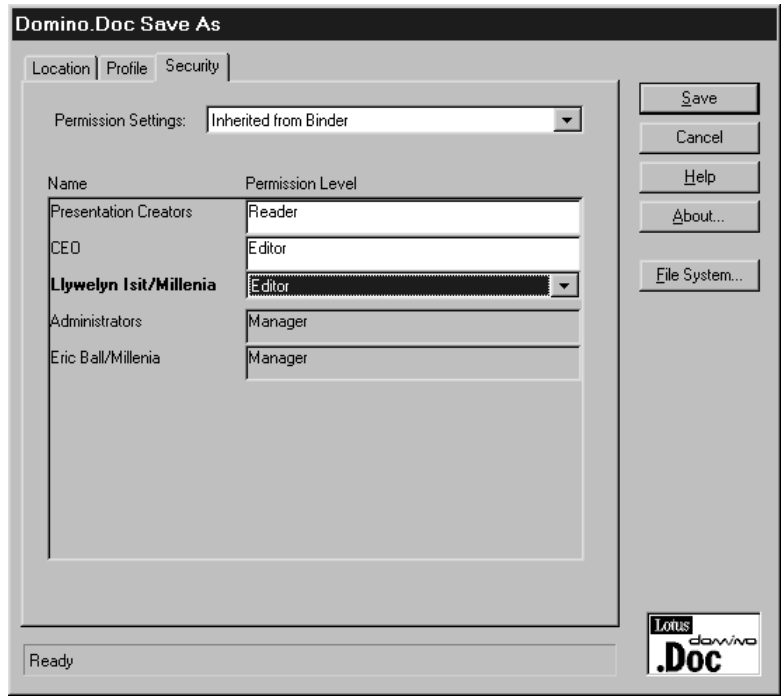
3. Browse to the binder where you want to store the document, and type the title of the document to be used in the Title field.



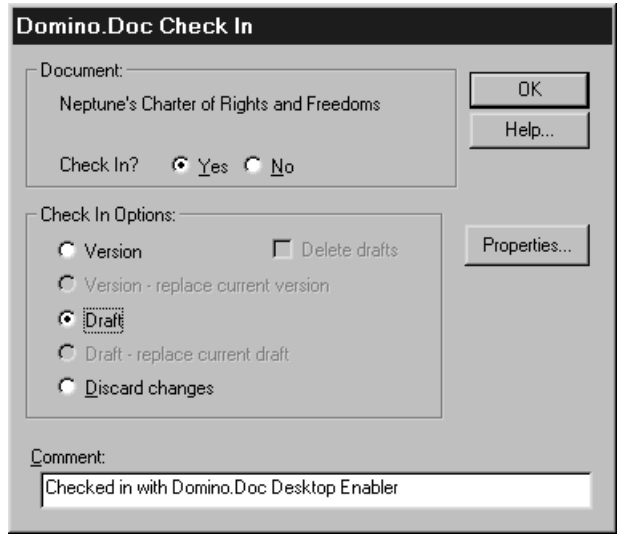
4. Click the Profile tab, and fill out the profile information as needed.



5. Click the Security tab and adjust the security as needed. Click the Save button to save it into the library.



6. Upon closing the document, you will be prompted whether you wish to check in the document. The following figure shows the check in option that you will see:



The default is to check in the document as a draft, but you can also keep it as your working copy or check it in as a version when you close it in your ODMA-enabled application.

Appendix H
Domino.Doc Storage Manager

Domino.Doc manages documents throughout their life cycle-from authoring through review, approval, distribution and archiving. The Domino.Doc Storage Manager plays a key role in the life cycle by providing automated, offline storage for documents. Through complete integration with Domino.Doc archival functionality, it moves documents from the Domino.Doc file cabinets to low cost media like jukeboxes or tapes, thereby freeing up disk space on the Domino.Doc server. When predefined archival criteria are met, for example, when a document's creation date is older than six months, the Domino.Doc Storage Manager automatically migrates the document out of the Domino.Doc file cabinet to an alternate repository. Domino.Doc retains the profile information about the archived documents, enabling users to later search and retrieve them. Frequently accessed or current documents are kept online for fast retrieval, while documents that are less frequently accessed or older can be moved to offline storage.

For organizations that need to manage a large volume of documents, Domino.Doc Storage Manager completes your Domino.Doc solution, by allowing you to optimally manage the available storage and minimize overall storage costs, while retaining enterprise-wide access to archived documents.

Following are the main advantages of Domino.Doc Storage Manager:

- Fully automated offline storage

  Complete integration with Domino.Doc allows documents to be archived automatically based on predefined criteria, without end user intervention.

- Manage more for less

  Reduce the volume of data stored on magnetic media, while retaining access to documents archived offline.

- Improve process efficiency

  Automated archiving ensures consistency in document storage procedures.

- Simple installation and configuration

  Full integration with Domino.Doc, coupled with robust, proven driver technology, lets you easily install and configure your offline storage solution.

- Intelligent archiving and re-archiving

  Archive and re-archive managed documents based on time, frequency of usage, availability of newer versions, or a custom formula.

- Optimize use of server disk space

  Domino.Doc Storage Manager automatically moves documents to less expensive storage media, thereby freeing up server disk space. Now you can establish an optimal balance between the cost of storing documents and the need for fast and frequent access.

The Domino.Doc Storage Manager consists of two components:

- The Connection for Domino.Doc Storage Manager
- The Storage Server for Domino.Doc Storage Manager

Storage Manager Connection

The Storage Manager Connection contains the archive and retrieve agents which reside in the Domino.Doc library and connect to the Storage Server. The Storage Server Connection must be installed on the Domino.Doc master library server. Archiving can occur from any master file cabinet, but cannot occur from a replica file cabinet. Retrieval requests can be made from any replica, but the information itself is first restored to the master file cabinet replica and through replication makes its way to the replica server from which the retrieval request originated.

Installing Storage Manager Connection will modify the Domino.Doc templates. It adds two agents to the file cabinet template and one subform to the library template. The agents are:

Archive to DDSM

This agent identifies the documents that are ready for archiving based on the archive criteria defined in the document type form, and sends these to the Storage Server. The Archive to DDSM agent is enabled to run in the background on a daily schedule.

The actual name of the agent appearing in the agent list will be the name of the Domino server on which the master library resides, dash (-), and Archive to DDSM. While specifying the agent name in the document type form, you should specify this full name.

Retrieve from DDSM

This agent retrieves archived documents from the Storage Server when a retrieval request is sent. The file attachment, which has been archived, is restored to the Domino.Doc document. Retrieve options are specified in the document type form. The Retrieve from DDSM agent is enabled to run in the background on a daily schedule.

The actual name of the agent appearing in the agent list will be the name of the Domino server on which the master library resides, dash (-), and Retrieve from DDSM. While specifying the agent name in the document type form, you should specify this full name.

The subform is:

DDSMArchiveParameters

This subform is used to specify the details of Storage Server like ADSM Server, Node Name, Password, ADSM TCPIP Port, and Management Class.

Storage Server

The Storage Server for Domino.Doc Storage Manager stores and retrieves Domino.Doc documents to and from the attached storage device. The Storage Server can be installed on the same physical server as Domino.Doc or on a separate server. For maximum performance, install Storage Server on a separate server from the Domino.Doc server.

Installing Storage Server does the following:

- Adds the server software for providing administrative server resources to Domino.Doc servers. It uses the storage management databases, recovery log, and server storage to ensure data availability. The Storage Server coordinates storage of documents to disk, optical disk, or tapes.
- Adds the utilities for configuring the Storage Server. The utilities that you will use include volume formatting for creating and modifying storage pool volumes, services, and server options.
- Adds the administrative client for controlling the Storage Server.
- Adds IBM ADSM online books including *IBM ADSTAR Distributed Storage Manager for Windows NT Administrator's Guide*.

In the above section, we provided a brief overview of Domino.Doc Storage Manager. But before using Domino.Doc Storage Manager to archive and retrieve documents, you should know more about how to install Storage Manager components, how to configure Storage Server (like defining storage objects and policy objects), and how to configure automated libraries. These topics are not within the scope of this redbook.

Important The current available version of Domino.Doc Storage Manager will not directly work with Domino.Doc 2.5. You have to first install Domino.Doc 2.1 on Domino 4.6 and install Storage Manager, then upgrade Domino.Doc to version 2.5 and Domino to version 5.

Appendix I
Domino.Doc Imaging Client

Domino.Doc Imaging Client provides desktop-based imaging, annotation, and optical character recognition (OCR) capabilities. Using the Imaging Client, you can scan paper documents or import images and store them in a Domino.Doc library. The document can then be viewed by any authorized person across your organization.

With Domino.Doc Imaging Client, images of paper documents become just another document type. Paper documents are often a mess in many organizations. Their storage and retrieval is a time-consuming process. It is more difficult and costly to maintain paper documents than electronic ones. Domino.Doc Imaging Client provides organizations with a cost-effective method for maintaining the vital document assets that remain a critical part of so many business processes.

The important features of the Domino.Doc Imaging Client are:

- Full integration with Domino.Doc

  Domino.Doc Imaging Client easily stores images in Domino.Doc file cabinets. Its familiar Windows interface enables anyone to work comfortably with image documents, without much training.

- Advanced annotation tools

  You can easily add your input to images using the annotation features like margin notes, highlighting, and pop-up annotation.

- Leverage more documents

  Capture paper documents in Domino.Doc for faster access, higher efficiency and greater business productivity.

- Ease of use

  Domino.Doc Imaging Client is a Windows-based desktop application that is fully integrated with Domino.Doc. To capture documents, just scan the paper document or import the image. Then easily store it in Domino.Doc.

- Built in OCR

  Domino.Doc Imaging Client uses OCR technology to turn paper-based documents into searchable, editable text.

- Support for standard file formats

  Domino.Doc supports all standard file formats including BMP, TIFF, GIF, JPEG, PCX, DCX and XIF formats.

Appendix J
Domino.Doc integration with Domino Workflow

Domino Workflow is a workflow development and management system for designing electronically driven processes such as customer orders, design reviews, and expense reports. Domino Workflow automates your daily business procedures and assists in coordinating tasks between work groups. Business processes are modeled based on the existing document flow, and are enhanced by the ability to modify them. Domino Workflow extends the existing Domino environment to handle both structured and ad hoc workflow processes.

Domino.Doc and Domino Workflow are a natural partnership. By integrating Domino.Doc with Domino Workflow, you can leverage all of the robust document management capabilities of Domino.Doc and the powerful workflow management capabilities of Domino Workflow. Store and manage documents throughout their life cycle within Domino.Doc, and then use Domino Workflow to design and manage processes to route those documents and your work within your enterprise. Documents created through Domino Workflow processes can be stored and managed in Domino.Doc, while document management applications requiring more structured workflow can leverage the capabilities of Domino Workflow.

Domino.Doc with Domino Workflow integrates the life cycle management of documents with the management of structured business processes:

- Domino.Doc provides life cycle management for the creation, review, approval, and archiving of documents.
- Domino Workflow provides a rich solution to address workflow management.

The integration of Domino.Doc with Domino Workflow means that:

- Domino Workflow applications that are based on documents and files can now leverage the capabilities of Domino.Doc.
- Domino.Doc applications that require more structured workflow can now leverage the capabilities of Domino workflow.

Integration results in added features in each product that provide important application points without added programming. For more advanced needs, each provides the ability for additional customization and integration.

In Domino.Doc these added features include:

- Starting workflow jobs from Domino.Doc

  Workflow jobs can be started from within Domino.Doc. This is accomplished by associating workflows with Domino.Doc events by using the workflow section in the document type definition form. When the specified event is completed, the associated workflow job starts automatically.

  Workflow jobs can also be started manually by the user. The list of processes that are available for such a manual initiation can also be specified in the document type definition of Domino.Doc documents.

- Finding related workflows from Domino.Doc

  A new tab has been added to Domino.Doc for Domino Workflow integration. Users can find workflow jobs associated with the current document.

In Domino Workflow these added features include:

- Linking Domino.Doc documents with Domino Workflow documents

  Based on ready to use design elements, Domino Workflow designers can easily create workflow documents that link to Domino.Doc documents. Users can view files stored in Domino.Doc directly from the workflow user interface. They can also switch back and forth between Domino Workflow and Domino.Doc to check out documents or to review older versions, for example.

- Searching documents in Domino.Doc for related Domino Workflow documents

  Depending on the context of a workflow job, such as a customer name or contract ID, you can let Domino Workflow search against Domino.Doc and make the results available to workflow participants.

- Controlling Domino.Doc documents from Domino Workflow

  You can let a workflow job take full control of a file handled in Domino.Doc. All other users can only read the document while it is being processed by a workflow. Intermediate drafts and versions can be preserved in Domino.Doc for later retrieval. After being processed, the document can be made available to Domino.Doc users again.

- Archiving Domino Workflow documents to Domino.Doc

  You can use Domino.Doc as an archive for documents that have been created or processed by a workflow job. This feature allows you to take advantage of the Domino.Doc document life cycle management and the advanced archiving technologies available in Domino.Doc.

Special notices

This publication is intended to help you create customized solutions with Domino.Doc 2.5.

The information in this publication is not intended as the specification of any programming interfaces that are provided by Lotus Domino. See the publications section of the announcement for Lotus Domino and related products for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM products, programs, or services may be used. Any functionally equivalent program that does not infringe on any IBM intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available subject to appropriate terms and conditions, including, in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendors, and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate

and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

This document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF, when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX
AS/400
DB2
IBM®
MQSeries
OS/2
OS/Warp

The following are trademarks of Lotus Development Corporation in the United States and/or other countries:

Domino.Doc
LotusScript®
Lotus SmartSuite®
Notes Mail®
NotesPump
NotesSQL
Lotus®

Lotus Domino
Lotus Notes®
RealTime Notes
SmartIcons®

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

SET and the SET logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product or service names may be the trademarks or service marks of others.

Additional Web material

Additional Web material is referenced in this book and can be found on the IBM Redbooks Web site. The material is:

File name	*Description*
domdoc.zip	Zipped Domino.Doc library template (domdoc.ntf) customized with the examples in this redbook.
filecab.zip	Zipped Domino.Doc file cabinet template (filecab.ntf) customized with the examples in this redbook.
thesaurus.zip	Zipped Domino database (Thesaurus.nsf) with synonyms used in example in Chapter 5: Customizing forms.
website.zip	Zipped Domino database (website.nsf) with agent used in example in Chapter 11: Agents.
images.zip	Various images, like new.gif used in the examples in this redbook.
vbapisample	Source for Visual Basic project used in example in Chapter 10: Domino.Doc API.
approachdb	Lotus Approach database used in example in Chapter 11: Agents.
doccreatesccam.zip	Self-running Lotus ScreenCam (MillenniaDocCreate.exe) that shows document creation in our case study.
searchsccam.zip	Self-running Lotus ScreenCam (MillenniaSearch.exe) that shows addition of synonyms to search in our case study.
apisamplesccam.zip	Self-running Lotus ScreenCam (MillenniaAPISample.exe) that shows import of document into Domino.Doc from Visual Basic application.
DDocClasses.prz	Lotus Freelance file with methods, properties and events for the classes in the Domino.Doc object model.

How to get the Web material

To get the Web material point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG245658

Alternatively, you can go to the redbooks Web site at:

http://www.redbooks.ibm.com

Select Additional Materials and open the file that corresponds with the redbook form number.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

International Technical Support Organization publications

For information on ordering these ITSO publications see, "How To Get ITSO Redbooks."

- *Lotus Notes 5.0: A Developers Handbook*, IBM form number SG24-5331, Lotus part number CC7EDNA
- *Developing Web Applications Using Lotus Notes Designer for Domino 4.6*, IBM form number SG24-2183, Lotus part number 12974
- *Lotus Notes 4.5: A Developers Handbook*, IBM form number SG24-4876, Lotus part number AA0425
- *Using VisualAge for Java to Develop Domino Applications*, IBM form number SG24-5424, Lotus part number CT6ENNA
- *Connecting Domino to the Enterprise Using Java*, IBM form number SG24-5425, Lotus part number CT6EMNA
- *Lotus Domino for AS/400: Integration with Enterprise Applications*, IBM form number SG24-5345, Lotus part number CT7BMNA
- *LotusScript for Visual Basic Programmers*, IBM form number SG24-4856, Lotus part number 12498
- *Lotus Solutions for the Enterprise, Volume 1. Lotus Notes: An Enterprise Application Platform*, IBM form number SG24-4837, Lotus part number 12968
- *Lotus Solutions for the Enterprise, Volume 2. Using DB2 in a Domino Environment*, IBM form number SG24-4918, Lotus part number CT69BNA
- *Lotus Solutions for the Enterprise, Volume 3. Using the IBM CICS Gateway for Lotus Notes*, IBM form number SG24-4512
- *Lotus Solutions for the Enterprise, Volume 4. Lotus Notes and the MQSeries Enterprise Integrator*, IBM form number SG24-2217, Lotus part number 12992
- *Lotus Solutions for the Enterprise, Volume 5. NotesPump, the Enterprise Data Mover*, IBM form number SG24-5255, Lotus part number CT69DNA
- *Enterprise Integration with Domino for S/390*, IBM form number SG24-5150

Other Lotus-related ITSO publications

The publications listed in this section may also be of interest:

- *A Roadmap for Deploying Domino in the Organization*, IBM form number SG24-5617, Lotus part number CT6P8NA
- *The Three Steps to Super.Human.Software: Compare, Coexist, Migrate; From Microsoft Exchange to Lotus Domino, Part One: Comparison*, IBM form number SG24-5614, Lotus part number CT7QTNA
- *The Three Steps to Super.Human.Software: Compare, Coexist, Migrate; From Microsoft Exchange to Lotus Domino, Part Two: Coexistence and Migration*, IBM form number SG24-5615, Lotus part number CT7QWNA
- *Lotus Notes and Domino R5.0 Security Infrastructure Revealed,* IBM form number SG24-5341, Lotus part number CT6TPNA
- *Lotus Notes and Domino: The Next Generation in Messaging. Moving from Microsoft Mail to Lotus Notes and Domino*, IBM form number SG24-5152, Lotus part number CT7SBNA
- *Eight Steps to a Successful Messaging Migration:A Planning Guide for Migrating to Lotus Notes and Domino*, IBM form number SG24-5335, Lotus part number CT6HINA
- *Deploying Domino in an S/390 Environment*, IBM form number SG24-2182, Lotus part number 12957
- *The Next Step in Messaging: Case Studies on Lotus cc:Mail to Lotus Domino and Lotus Notes*, IBM form number SG24-5100, Lotus part number 12992
- *Lotus Notes and Domino: The Next Generation in Messaging. Moving from Novell GroupWise to Lotus Notes and Domino*, IBM form number SG24-5321, Lotus part number CT7NNNA
- *High Availability and Scalability with Domino Clustering and Partitioning on Windows NT*, IBM form number SG24-5141, Lotus part number CT6XMIE
- *From Client/Server to Network Computing, A Migration to Domino*, IBM form number SG24-5087, Lotus part number CT699NA
- *Netfinity and Domino R5.0 Integration Guide*, IBM form number SG24-5313, Lotus part number CT7BKNA
- *Lotus Domino R5 for IBM RS/6000*, IBM form number SG24-5138, Lotus part number CT7BHNA
- *Lotus Domino Release 4.6 on IBM RS/6000: Installation, Customization and Administration*, IBM form number SG24-4694, Lotus part number 12969
- *High Availability and Scalability with Domino Clustering and Partitioning on AIX*, IBM form number SG24-5163, Lotus part number CT7J0NA
- *Lotus Domino for AS/400: Installation, Customization and Administration*, IBM form number SG24-5181, Lotus part number AA0964

- *Lotus Domino for S/390 Release 4.6: Installation, Customization & Administration*, IBM form number SG24-2083
- *Lotus Domino for S/390 Performance Tuning and Capacity Planning*, IBM form number SG24-5149, Lotus part number CT6XNIE
- *Porting C Applications to Lotus Domino on S/390*, IBM form number SG24-2092, Lotus part number AB1720
- *Managing Domino/Notes with Tivoli Manager for Domino, Enterprise Edition, Version 1.5*, IBM form number SG24-2104
- *Measuring Lotus Notes Response Times with Tivoli's ARM Agents*, IBM form number SG24-4787, Lotus part number CT6UKIE
- *Using ADSM to Back Up Lotus Notes*, IBM form number SG24-4534

Redbooks on CD-ROMs

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at **`http://www.redbooks.ibm.com/`** for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	*Collection Kit Number*
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
Application Development Redbooks Collection	SK2T-8037
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
AS/400 Redbooks Collection	SK2T-2849
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Networking and Systems Management Redbooks Collection	SK2T-6022
System/390 Redbooks Collection	SK2T-2177
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

How to get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site `http://www.redbooks.ibm.com`**
  Search for, view, download or order hardcopy/CD-ROM redbooks from the redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this redbooks site.

  Redpieces are redbooks in progress; not all redpieces become redbooks and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.
- **E-mail Orders**
  Send orders by e-mail including information from the redbooks fax order form to:

	e-mail address
In United States:	usib6fpl@ibmmail.com
Outside North America:	Contact information is in the "How to Order" section at this site: **`http://www.elink.ibmlink.ibm.com/pbl/pbl/`**

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the "How to Order" section at this site: **`http://www.elink.ibmlink.ibm.com/pbl/pbl/`**

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada (toll free)	1-800-267-4455
Outside North America	Fax phone number is in the "How to Order" section at this site: **`http://www.elink.ibmlink.ibm.com/pbl/pbl/`**

The latest information for customers may be found at the redbooks Web site.

IBM intranet for employees

IBM employees may register for information on workshops, residencies, and redbooks by accessing the IBM intranet Web site at **`http://w3.itso.ibm.com/`** and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access **`MyNews`** at **`http://w3.ibm.com/`** for redbook, residency, and workshop announcements.

IBM redbook fax order form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

❑ Invoice to customer number

❑ Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, MasterCard, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Index

Symbols

@Command, 38
@DBColumn, 29
@DBLookup, 29
@DbManager, 38
@DbName, 38
@DbTitle, 38
@DDEExecute, 38
@DDEInitiate, 38
@DDEPoke, 38
@DDETerminate, 38
@DialogBox, 38
@formulas, 189
@PickList, 38
@PostedCommand, 38
@Prompt, 38
@ViewTitle, 38

A

Access Control List, 19, 186, 187
Add Search, 189, 192, 193
Add SearchIndex#, 189, 192, 193
ADSM, 38, 263
Agent List, 188
Agent Restrictions, 186
Annotation, 1, 265
API, 18, 139–145, 147–155, 157–162, 165–167, 171, 176–177, 184
Applets, 109
Applying, 43
  binder type, 49
  document type, 43
Approval, 3, 113, 267
  approvers, 4
  options, 34
  parameters, 34
  predefined, 123
Architecture, 5
  distributed, 10
Archival criteria, 261
Archive to DDSM, 262
Archiving, 19, 114, 268
  agent, 37
  options, 36
Authenticate Web users, 4
Authoring, 114
Automate tasks, 185
Automated tasks, 261

B

Background processing, 18
Binder, 6
  database, 7
  type, 7, 16, 46
Binder object, 141, 151, 153, 157, 181, 182
Binder TOC, 52, 80, 83, 85, 86, 212
Binders collection, 149, 151, 153, 155, 158, 163, 166, 180, 181

C

Cabinet
  object, 141, 147, 149, 159, 179, 180
Cabinets
  collection, 140, 149, 150, 152, 155, 158, 163, 166, 179
  property, 147
Case study, 19
Categorized, 48
  binder, 48
Check in, 3, 115
Check out, 3
CheckedOutDocuments
  property, 161
CheckIn method, 151, 154, 156, 164
Clustered server, 10
Collection class, 146, 147, 148
Companion products, 19
Compatibity with common Internet standards, 20
Compliance with ODMA standards, 3
Computed fields, 28
Configuration database, 6
Content management, 1
Count property, 146, 148, 150
Creating
  binder type, 48
  document type, 29
Custom agents, 19
Custom processing, 13, 15
Cycle
  approval, 3, 114, 120
  review, 3, 114

D

Databases, 6, 79, 92, 93, 217, 221
  Binder, 7
  Configuration, 6
  Document, 7
  File Cabinet, 6
  Library, 6
  Log, 6
  replication, 16
  Site Admin database, 6
  Transaction database, 6
DDSMArchiveParameters, 38, 263
DeleteAllRevisions method, 165
Deleting, 54
  binder type, 54
  document type, 54
Desktop applications, 1, 13
Desktop Enabler, 1, 14
Document, 6, 58–61, 65, 72, 74, 79, 80, 84, 86, 87, 89, 90, 98, 102, 108, 110, 111, 205, 207, 210, 212, 219, 220–222
  database, 7
  drafts, 115, 120
  type, 7, 16, 25, 265
  version, 115, 121
Document flow, 21
Document Imaging, 15
Document management, 267
Document object, 141, 154, 155, 156, 157, 159, 161, 163–166, 182, 183
Documents collection, 154, 155, 157–159, 163, 166, 182
Domino Workflow, 2, 41
Domino.Doc
  architecture, 5
  capabilities, 3

Domino.Doc Agents, 185, 190
Domino.Doc API, 139–142, 145, 147, 149, 151, 154, 162, 165, 167, 171, 184
Domino.Doc Desktop Enabler, 94, 140
Draft editors, 115
Drafts, 32, 115, 120

E

Electronic documents, 3
Embedded view, 90, 91
Error logging, 16
Events handling, 18
External application, 13, 15

F

FavoriteDocuments collection, 161
FavoriteDocuments property, 160
Field Attributes, 30
Field object, 141
Field type considerations, 28
File Cabinet, 3, 6, 62, 66, 68–70, 77, 79, 80, 83, 85, 87, 90, 95, 96, 101, 106, 108, 111, 203, 204, 208, 211, 213, 217, 218, 225
  database, 6
  template, 7
File Room, 3, 6
File servers, 3
Forms, 57, 58, 62, 66–69, 71, 93, 96, 106, 109, 203, 204
Framesets, 109, 111
Full text indexing, 16

G

GetDocumentById method, 159

H

HTML, 189
Hyper Text Transfer Protocol, 141–145, 148, 150, 152, 155, 158, 160, 163, 166, 177

I

Image resource, 109
Images, 1
Imaging, 1, 265
Imaging Client, 1
Imported Java, 189
Imported JavaIndex#, 185
Importing, 18
Improving process efficiency, 261
Integration, 13, 265
Integrity, 4
Intranet, 3
Item property, 146, 147
ItemByIndex property, 146
ItemByName property, 146
ItemByTitle property, 146
ItemByValue property, 146, 147, 149

J

Java, 185, 186, 187, 189
JavaIndex#, 185, 186
Jukeboxes, 1, 261

K

Knowledge management, 1, 22

L

laction.gif file, 98, 102
Library, 1, 6, 58–60, 66, 67, 69–71, 79, 80, 85, 93, 94, 96, 101, 106, 108, 110, 111, 203, 207, 208, 210, 215–217
  database, 6
  replica, 7, 126
  template, 6
Library class, 141
Library object, 140–145, 147–150, 152, 155, 158–162, 166, 177
Life cycle, 1, 16, 113, 261, 267
Log database, 6
Lotus Approach, 195, 199
Lotus Notes Automation classes, 173
LotusScript, 140, 142, 143, 156, 159, 164, 185–187, 189, 197

M

Mail routing, 10
Manageability, 21
Management Class, 38
Master library server, 262
Millennia Space Travel Corporation, 139, 141, 167, 171, 185, 190, 195, 202
Modifying, 54
  forms, 17
  navigators, 17
  subforms, 17
  views, 17

N

Name property, 146
Navigators, 93–96, 99, 101, 106, 109, 215, 217
Notes client, 142, 171
Notes formula, 185
Notes protocol, 141, 142
Notes simple action, 185
NotesSearch, 58, 66–68, 71, 75, 77, 108, 111, 203, 204
NotesSearch and Sample Notes App Integration, 58
Notifying, 19

O

OCR, 1, 265
ODBC, 196
ODMA, 1, 14, 19
Offline, 1
  storage, 261
Options
  approval, 34
  archiving, 36
  retrieve, 39
  review, 32
  version, 30
  workflow, 41
Outlines, 71, 109, 111
Overview, 1
  Domino.Doc, 1
  replication, 9

P

Pages, 109, 111
Profile, 25
Profile object, 141
Programmer's pane, 77
Properties, 62, 76, 82, 83, 86, 89, 90, 92, 98, 102, 103, 106, 107

Q

Query, 78, 210
Quick and Easy, 7

R

R5 design elements, 94, 109
RDBMS, 185
Reasons to customize, 13
Rebuilding, 52
Replication, 3, 9
  conflicts, 9
Reports, 18
Repository, 3, 261
Requirements, 19, 20
ResolveLink method, 161
Resources, 109, 110, 111
Retrieval agent, 39
Retrieval Options, 39
Retrieve from DDSM, 263
Reverse Slashes function, 105
Review, 3, 113, 267
  parameters, 32
  predefined, 117
  reviewers, 4
  reviewing, 115
Review Options, 32
Room object, 140, 147, 178, 179
Rooms collection, 147, 148, 177, 178, 179
Rooms property, 147
Routing type, 33, 35
Run On field, 192, 197

S

Sample Notes App Integration, 58, 203
Scenario, 20
Script Libraries, 109
Search, 58, 66, 67, 69, 70–74, 78, 108, 111, 210
  searching, 4, 268
SearchDocumentByID method, 159
Security, 4, 185, 186, 202
Security object, 141
SetContents method, 154
SetToReviewCopy method, 161
SetToWorkingCopy method, 161
Setup
  approval, 121
  review, 115
Simple Actions, 185, 189, 194
Single category view, 90
Storage management, 1
Storage Manager, 1, 36
Storage Server, 263
Subform, 26, 38, 46, 57, 109
  binder type, 48
  design, 26, 46
  document type, 26
  modifying, 54
Synonym search, 71
System Profile, 95
System requirements, 5

T

Task automation, 202
Template, 62, 66, 77, 80, 81, 83, 85, 87, 90, 92, 94, 95, 96, 101, 106, 203, 204, 205, 207, 212, 262
  file cabinet, 7, 8
  library, 6, 8
Third party technology, 13, 15
Threshold limit, 219, 220, 221
Time limit, 33, 35
Title property, 146, 151, 153, 154
Topology, 11
Transaction database, 6
Transaction Manager, 7
Transactions, 16

U

URL, 143, 144, 145, 148, 150, 152, 155, 158, 160, 162, 166, 177

V

Version control, 3
Version Options, 30
Views, 62, 65, 72, 74, 79–81, 83–93, 108, 110, 111, 207–213
Visual Basic, 139, 140, 142, 143, 147–150, 167, 171, 172
Visual C++, 140, 142, 144

W

Web, 59–61, 66, 68–70, 79, 80, 93, 95, 96, 99, 102, 106, 109, 110, 209, 210, 213
  browser, 1, 171
  client, 141
  site, 185, 190, 195, 196
WebQueryOpen, 189
WebQuerySave, 189
WebSimpleSearchAll:, 66
Workflow, 2, 3, 267
  ad hoc, 267
  jobs, 268
  options, 41
  parameters, 41

ITSO redbook evaluation

Creating Customized Solutions with Domino.Doc SG24-5658-00

Your feedback is very important to help us maintain the quality of ITSO redbooks.

Please complete this questionnaire and return it using one of the following methods:

- Use the online evaluation form at **`http://www.redbooks.ibm.com/`**
- Fax it to: USA International Access Code +1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?

___Customer ___Business Partner ___Solution Developer ___IBM employee

___None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes _____ No _____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____

_____

_____

_____

Printed in the U.S.A.

SG24-5658-00

Part No. CC6X3NA

Creating Customized Solutions with Domino.Doc

SG24-5658-00